IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

Sandpiper CDN, LLC, *Plaintiff*, v. Comcast Cable Communications, LLC d/b/a Xfinity; Comcast Cable Communications Management, LLC d/b/a/ Comcast Technology Solutions; and Comcast Corporation. *Defendants*.	Civil Case No. _____ **JURY TRIAL DEMANDED**

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Sandpiper CDN, LLC ("Sandpiper CDN," "Sandpiper," or "Plaintiff") hereby files this Complaint for patent infringement against Comcast Cable Communications, LLC d/b/a/ Xfinity, Comcast Cable Communications Management, LLC d/b/a Comcast Technology Solutions, and Comcast Corporation ("Comcast" or "Defendants"), and alleges as follows:

NATURE OF THE ACTION

1. This is a civil action against Comcast for patent infringement arising under the United States Patent Laws, 35 U.S.C. § 271, *et. seq.* for the infringement of United States Patent No. 7,013,322, U.S. Patent No. 8,478,903, U.S. Patent No. 9,628,347, U.S. Patent No. 9,660,876, and U.S. Patent No. 9,762,692 (collectively "the Asserted Patents"). A true and correct copy of each Asserted Patent is attached to this Complaint as Exhibits A-E, respectively. Each of the Asserted Patents is owned by Plaintiff Sandpiper CDN, and Plaintiff and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre- and post-filing damages for

the full period allowed by law for infringement of the Asserted Patents, including compliance with 35 U.S.C. § 287.

PARTIES

2. Plaintiff Sandpiper CDN is a Delaware limited liability company with its principal place of business in Wilmington, Delaware.

3. Defendant Comcast Cable Communications, LLC is a limited liability company organized and existing under the laws of the State of Delaware, having its principal place of business at One Comcast Center, 1701 John F. Kennedy Boulevard, Philadelphia, Pennsylvania 19103. Comcast Cable Communications, LLC may be served through its registered agent Comcast Capital Corporation, 1201 N. Market Street, Suite 1000, Wilmington, Delaware 19801.

4. Defendant Comcast Cable Communications Management, LLC d/b/a Comcast Technology Solutions[1] is a limited liability company organized and existing under the laws of the State of Delaware, having its principal place of business at One Comcast Center, 1701 John F. Kennedy Boulevard, Philadelphia, Pennsylvania 19103. On information and belief, Comcast Cable Communications Management, LLC has been registered to do business in the State of Texas since at least November 10, 2011, and may be served through its registered agent Corporation Service Company d/b/a CSC-Lawyers Inc., 211 E. 7th Street, Suite 620, Austin, Texas 78701.

5. Defendant Comcast Corporation is a Pennsylvania corporation with its principal place of business at One Comcast Center, 1701 John F. Kennedy Boulevard, Philadelphia, Pennsylvania 19103. On information and belief, Comcast Corporation has been registered to do business in the State of Texas since at least November 30, 2018, and may be served through its

[1] *See* Dkt. 1, *The Teaching Company, LLC et al. v. Sling TV L.L.C. and Dish Techs. L.L.C.*, Case: 1:21-cv-740 (D. Colo. Mar. 11, 2021) (identifying Comcast Technology Solutions as a d/b/a of Comcast Cable Communications Management, LLC in declaratory judgment complaint).

registered agent at Corporation Service Company d/b/a CSC-Lawyers Inc., 211 E. 7th Street, Suite 620, Austin, Texas 78701.

JURISDICTION AND VENUE

6. This action arises under the United States Patent Laws, Title 35 of the United States Code. This Court has subject matter jurisdiction over this action under 28 U.S.C. §§ 1331 and 1338(a).

7. This Court has personal jurisdiction over Comcast in this action because Comcast has committed acts within the Eastern District of Texas giving rise to this action and has established minimum contacts with this forum such that the exercise of jurisdiction over Comcast would not offend traditional notions of fair play and substantial justice. Comcast, directly and/or through subsidiaries or intermediaries, has engaged in continuous, systematic, and substantial activities within this State, including substantial marketing and sales of products—including Comcast products using the accused Comcast CDN functionalities[2]—within this State. In addition, Comcast, directly and/or through subsidiaries or intermediaries, committed and continues to commit acts of infringement in this District by, among other things, leveraging the offices, facilities, and/or employees Comcast maintains in this District and the State of Texas to make, use, sell, and offer to sell the accused Comcast CDN functionalities, and thereby actively directing its activities to customers located in the State of Texas.

8. This Court also has jurisdiction over Comcast in accordance with due process and/or the Texas Long Arm Statute because, in part, Comcast conducts business in the State of Texas by "recruit[ing] Texas residents, directly or through an intermediary located in this state, for employment inside or outside this state." TEX. CIV. PRAC. & REM. CODE § 17.042(3).

[2] The "accused Comcast CDN functionalities" are defined herein at ¶¶ 26-33, 61 *et seq.*, *infra*.

9. Venue is proper in this District pursuant to 28 U.S.C. §§ 1391(b) and (c) and/or 1400(b). Defendants maintain a regular and established place of business in the State of Texas, transact business in the Eastern District of Texas, and committed and continue to commit acts of patent infringement in the Eastern District of Texas.

10. Comcast maintains a permanent physical presence within the Eastern District of Texas, conducting business from numerous locations, including at least 2740 N. Dallas Parkway, Suite 100, Plano, Texas, 75093[3] and 3500 E. Plano Parkway, Plano, Texas, 75074. Further, Defendants Comcast Corporation and Comcast Cable Communications Management, LLC are registered to conduct business in the State of Texas and has appointed Corporation Service Company d/b/a CSC-Lawyers Inc., 211 E. 7th Street, Suite 620, Austin, Texas 78701 as its agent for service of process.

11. On information and belief, Comcast also maintains a Texas office at 6200 Bridge Point Parkway, Suite 500, Austin, Texas 78730, where offerings utilizing the accused Comcast CDN technology are developed. In addition, on information and belief, Comcast employees, either directly and/or through subsidiaries or intermediaries, involved in the design, maintenance, and engineering of product and services utilizing the accused Comcast CDN technology, currently reside in Texas. Further, as of October 31, 2024, Comcast had job postings for 19 different positions in Texas, including jobs pertaining to video delivery infrastructure, engineering, and development.

12. Comcast directly and/or through subsidiaries or intermediaries tests, distributes, markets, offers to sell, sells, and/or utilizes products and services utilizing the accused Comcast

[3] *See Comcast Business Closes Masergy Acquisition*, Comcast Corporate Website (last visited Oct. 2, 2024), available at: https://corporate.comcast.com/press/releases/comcast-business-closes-masergy-acquisition.

CDN technology in the Eastern District of Texas, and otherwise purposefully directs infringing activities to this District in connection with its products and services utilizing the accused Comcast CDN technology. In addition, on information and belief, Comcast also employs a leadership team dedicated specifically to the State of Texas,[4] and invested "more than $100M" in 2023 to expand its Xfinity network and product offerings in Texas, with plans to continue expanding its network across the State "in a major way in 2024."[5]

FACTUAL BACKGROUND

Content Delivery Networks

13. Today, content delivery networks ("CDN") provide the critical services that enable content providers to quickly deliver online content to millions of consumers simultaneously over the Internet. But this has not always been the case.

14. In the early 1990s, the World Wide Web saw increasing adoption, becoming a household staple. This mass adoption led to data congestion issues due to the ever-growing number of users seeking to simultaneously access Internet content. A typical computer server in the 1990s, for example, could only handle a limited number of simultaneous connections before becoming overloaded. Moreover, signals take time to move through physical internet cables, and consumers living far from the physical server(s) hosting content experienced sluggish load times and high latency due to problems such as overloaded servers, congested network segments, and geographic separation.

[4] *See Leadership Team, Comcast Texas*, https://texas.comcast.com/leadership-team/ (last accessed October 28, 2024).
[5] *See Expansion, Comcast Texas*, https://texas.comcast.com/expansion/ (last accessed October 28, 2024).

Sandpiper Networks

15. In the mid-1990s, Andrew Swart and David Farber were among the first individuals to develop services that allowed content providers to distribute their content over the Internet, while avoiding the common congestion and performance issues that plagued Internet transmission at that time. One solution developed by Mr. Swart and Mr. Faber was to deploy CDN servers around the world. In doing so, Mr. Swart and Mr. Faber developed infrastructure and logic to replicate appropriate content from customers' origin servers to appropriate CDN servers, transparently rendezvous end users requesting that content to the "best" CDN server to deliver that content, while providing customers with control over their content and user experience. This service and its architecture was quickly imitated by many others in the industry, including Comcast.

16. Mr. Swart and Mr. Farber developed technology that empowered consumers to connect to an edge server that was closer to them and that had available bandwidth. This revolutionary infrastructure and logic provided numerous technical benefits. For example, distributing content across a network of servers alleviated data congestion issues. Additionally, allowing consumers to connect to nearby edge servers, instead of distant origin servers, reduced latency. Mr. Swart and Mr. Farber developed and built systems and methods for propagating data from origin servers to edge servers (in one example, a process known as "caching") based on network demand and for seamlessly routing users to the optimal edge server with the correct content.

17. In 1996, Mr. Swart and Mr. Farber founded Sandpiper Networks Inc. to further develop and commercialize their novel concept for a CDN. Sandpiper Networks was based in Thousand Oaks, California. Beginning in 1996, Sandpiper Networks designed and built a CDN

referred to as "Footprint." By at least May 24, 1996, the Sandpiper team developed infrastructure for delivering streaming resources, such as audio and video, using Sandpiper's CDN.

18. Sandpiper Networks labored not only to build and implement its CDN, but also to protect their groundbreaking innovation through patent protection. Recognizing that its invention could revolutionize content delivery worldwide, Sandpiper Networks filed numerous patent applications directed to its foundational CDN technology.

19. By at least May 1998, Sandpiper Networks was caching content and delivering cached content to end users of content providers using its CDN. Sandpiper Networks' first paying customer, the L.A. Times, paid Sandpiper Networks to host the report of Independent Counsel Ken Starr on his investigation of President Bill Clinton ("the Starr Report") beginning on September 11, 1998. Sandpiper's CDN was capable of caching, and used to cache and deliver Internet resources including *inter alia*, pictures, text files, dynamic resources, and streaming multimedia resources.

20. By October 30, 1998, Sandpiper partnered with WebRadio to utilize Sandpiper's CDN to deliver streaming audio from radio stations on behalf of WebRadio. This streaming audio was readily available to any Internet user.

21. On April 19, 1999, Sandpiper used its CDN to broadcast a live concert by the band "Big Bad Voodoo Daddy."

22. In December 1999, Sandpiper merged with Digital Island, Inc. ("Digital Island"), which filed additional patent applications directed to CDN technology.

23. In January 2007 and following a series of acquisitions, the assets of Sandpiper Networks and Digital Island, as well as their CDNs and patents, were acquired by Level 3.

24. Following its acquisition of Sandpiper's CDN, Level 3 understood the

groundbreaking technologies pioneered by Sandpiper Networks. Level 3 continued building upon the foundation laid by Sandpiper, eventually becoming one of the foremost CDN operators in the United States.

Industry Infringement

25. In the early-to-mid 2000s, demand for CDNs exploded. This increased demand prompted a slew of companies to enter the CDN market. These companies commercialized their own CDNs that incorporated the foundational CDN technology pioneered and patented by Sandpiper Networks and Level 3. In doing so, these companies capitalized on the investment made into CDN research and development made by Level 3 and/or its predecessors, misappropriating years of research and investments.

Comcast's Infringing Services

26. One such company is Comcast. By 2010, Comcast's business model developed a need for a central network to deliver content to customers across Comcast's networks. Leveraging the success of CDN technology pioneered by Level 3 and/or its predecessors, Comcast built its own in-house CDN to provide its customers access to a large library of content through live television networks, applications, websites, and other video-on-demand streaming services.[6] By leveraging technology of the Asserted Patents, and on information and belief, Comcast's in-house CDN delivered quick download speeds, optimized media delivery, experienced less buffering, and delivered an improved customer experience, among other technical benefits touted by Comcast, as shown in the screenshots below.

[6] https://corporate.comcast.com/comcast-voices/the-evolution-of-the-comcast-content-delivery-network

With our CDN solutions, your media will be delivered at broadcast quality without sacrificing download speed.

https://www.comcasttechnologysolutions.com/content-delivery-network-cdn-solutions

Content owners and distributors understand that in order to remain competitive, delivery strategies need to evolve faster than the insatiable appetite for streaming video, online gaming, and large media downloads. Today, the consumer experience with online content should be at least as good as — if not better than — that of traditional broadcast video. Comcast Technology Solutions solves for this with a tailored content delivery network architecture specifically designed to deliver online media at scale. Extraordinary experiences delivered to every screen must be consistent and reliable. We can get you there.

https://www.comcasttechnologysolutions.com/content-delivery-network-cdn-solutions

Optimized for media delivery.

Quality of experience drives consumer engagement and consumers won't wait for buffering, bad streams, or slow downloads. The Comcast CDN addresses these challenges by offering faster start times, faster downloads, higher resolutions, less buffering, fewer failures and higher overall completion rates for content and technology providers who deliver any type of online media.

Powered by the Comcast network.

Hundreds of edge caches throughout the domestic United States - including in the largest metro areas - enable distribution of content to millions of broadband users. These caches are powered by the largest fully converged network in North America supporting an Nx100G fiber optic backbone, millions of Wi-Fi hot spots, and robust interconnects with Tier 1 networks globally.

https://cdnportal.comcast.net/#/

> However, some CDNs are better equipped to handle these spikes. The Comcast CDN was purpose-built for large media delivery at the highest scale for live events. With more than 100 nodes and one of the largest, most advanced networks in North America, the Comcast CDN easily manages demand spikes while maintaining stream and download quality.

> If you have media content that is prone to spikes such as movies, TV shows or online games, you will want a CDN designed to protect your content and customers. The Comcast CDN, available from Comcast Technology Solutions, offers the most complete solution that can be used as a stand-alone CDN or as part of your multi-CDN strategy.

https://www.comcasttechnologysolutions.com/node/761

27. On information and belief, Comcast initially developed "Traffic Control" as its internal CDN solution, and subsequently released it as open source software in April 2015. The Apache Software Foundation accepted the Traffic Control open source software in 2016, where it subsequently became a top level project and officially renamed Apache Traffic Control ("ATC"). Comcast uses Apache Traffic Control "for all its IP video delivery, delivering images and software to its X1 platform, and for delivering third party content to its footprint."[7]

28. In one example leveraging ATC as its in-house CDN, Comcast employs a CDN architecture specifically designed to consistently and reliably deliver content at scale.[8] Comcast's CDN, and services relying on that CDN, use technology described and claimed by the Asserted Patents to meet its enormous data streaming needs. One example of Comcast's CDN architecture is provided below, which includes a Traffic Server infrastructure, a Traffic Control infrastructure, a DNS server, and/or a client, among other components.

[7] https://traffic-control-cdn.readthedocs.io/en/v7.0.1/faq.html.
[8] https://www.comcasttechnologysolutions.com/sites/default/files/2020-01/Comcast-CDN_OneSheet_01092020.pdf.

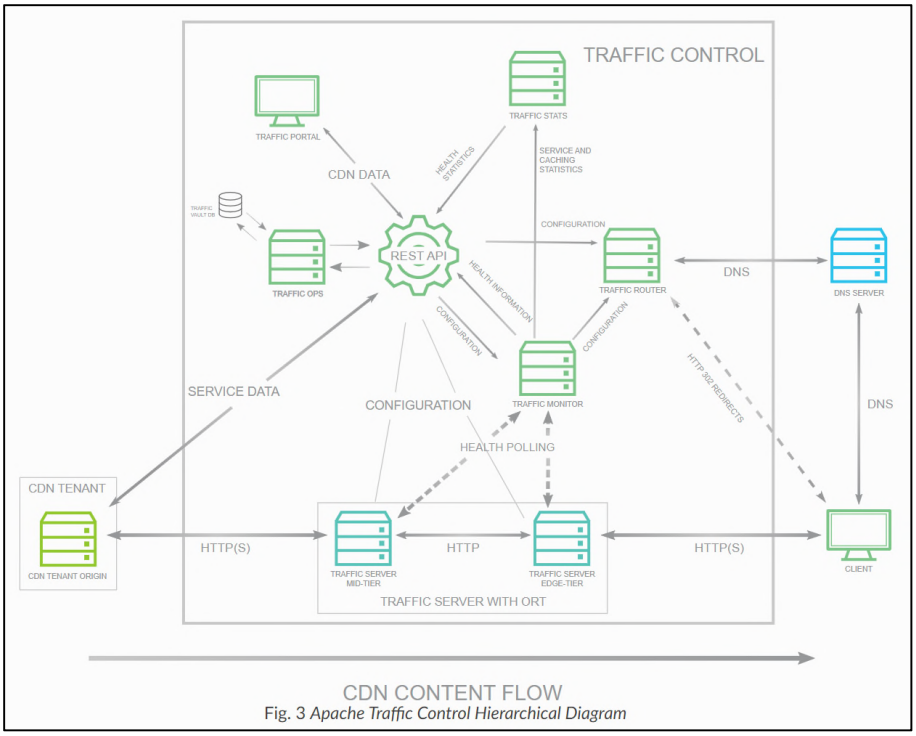
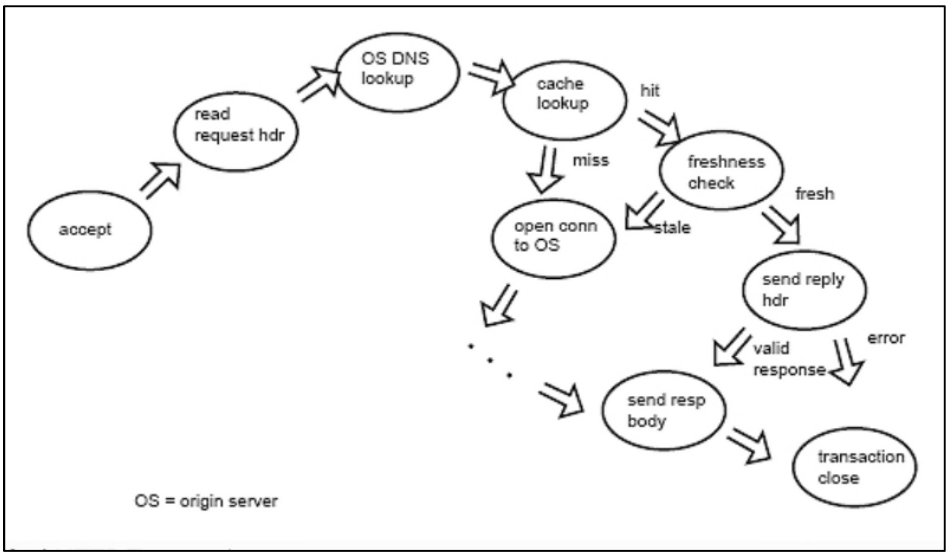


Fig. 3 *Apache Traffic Control Hierarchical Diagram*

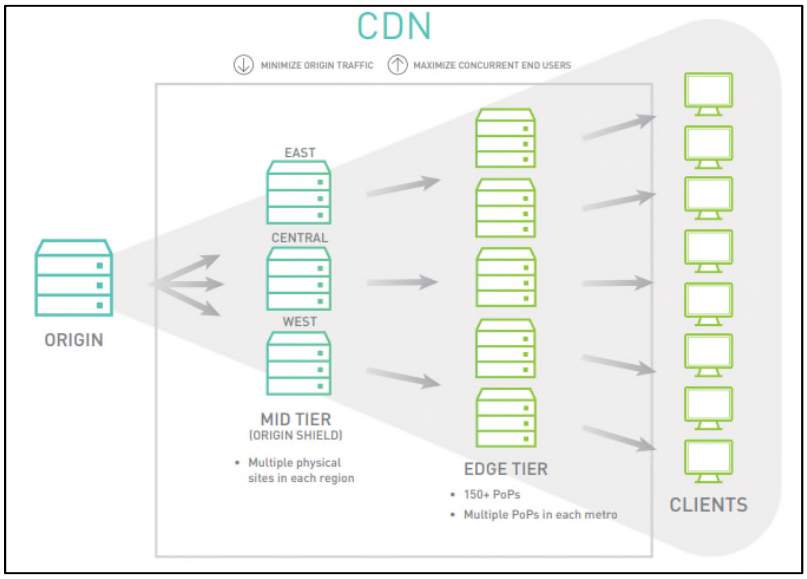
https://traffic-control-cdn.readthedocs.io/en/latest/overview/introduction.html

29. Moreover, ATC provides tools for customizing or extending the capabilities of its CDNs. For example, ATC provides a library of plugins, which empower Comcast to personalize its CDNs and achieve certain technical benefits provided by the Asserted Patents.



https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html

30. In one example, and on information and belief, Comcast employs a tier of servers, leveraging ATC as its in-house CDN. For example, Comcast CDN, organized between an origin server and its clients, includes any number of tiers. As illustrated in the example below, Comcast's CDN includes a mid-tier assembly and an edge tier assembly. In this manner, content can be better distributed within Comcast's CDN to achieve the technical benefits provided by the Asserted Patents.



THE COMCAST CDN: MORE THAN JUST "A BIGGER NET"

Comcast Technology Solutions offers one of the most broadly deployed caching platforms in North America, comprised of more than 150 physical locations. The extensive scale and reach of the Comcast CDN can readily serve as your primary delivery network or augment other content delivery providers. The open architecture and flexible commercial terms allow it to easily tie into a multi-CDN environment, aligned to your specific objectives and geographical needs.

MEET YOUR AUDIENCE ON THEIR TERMS

When your content is delivered through the Comcast CDN, you take advantage of the very same capabilities that power Comcast's entire X1 experience. Our broad distribution of caches in the largest metro areas provides multiple network paths to mitigate congestion, a tiered architecture to ensure scalability, and full support for live and on demand content across any device, anywhere. You can confidently present an extraordinary experience to your audience, and keep your focus on making great content.

https://www.comcasttechnologysolutions.com/sites/default/files/2020-01/Comcast-CDN_OneSheet_01092020.pdf

31. On information and belief, Comcast further leveraged other software tools made available through the Apache Software Foundation. One such tool is Apache Kafka. The Apache Software Foundation provides that Apache Kafka "distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications."[9] On information and belief, Comcast is one of these companies. As described in detail below, and on information and belief, Comcast employs Apache Kafka to efficiently update CDNs based on event streams as claimed by the Asserted Patents.

32. Comcast touts that its CDN "offers one of the most broadly deployed catching platforms in North America, comprised of more than 150 physical locations."[10] Leveraging this CDN technology enables Comcast to save on costs related to Comcast's IP video delivery,[11] and generate billions of dollars in revenue across several product segments.

33. For instance, Comcast's CDN technology is used and sold by Comcast, directly and/or through subsidiaries or intermediaries, in connection with services such as Comcast's X1 Comcast's Xfinity X1 platform, Comcast's media segment products and services (including NBCUniversal[12] broadcast networks and stations (*e.g.*, NBC Sports, NBCOlympics.com, Peacock, and NBC affiliates), the Peacock streaming service, and Sky-branded broadcast networks), and Comcast's business solutions (including Comcast's CTSuite Services). The Comcast CDN technology and architecture, and the services Comcast offers for sale that utilize this technology

[9] https://kafka.apache.org/

[10] https://www.comcasttechnologysolutions.com/sites/default/files/2020-01/Comcast-CDN_OneSheet_01092020.pdf.

[11] https://traffic-control-cdn.readthedocs.io/en/latest/faq.html.

[12] NBCUniversal Media, LLC is owned by Defendant Comcast Corporation. *See* Corporate Disclosure Statement, Dkt. No. 88, *UnoWeb Virtual, LLC v. eBay, Inc. et al.*, No. 2:16-cv-122-JRG (E.D. Tex. 2016).

and architecture, are collectively referred to herein as the accused "Comcast CDN" functionalities. Comcast did not license this patented technology from Sandpiper CDN, Level 3 and/or its predecessors.

34. In early 2024, Level 3 sold the Asserted Patents to Plaintiff Sandpiper CDN. Sandpiper CDN is now the owner in right, title, and interest in and to each of the Asserted Patents.

Sandpiper CDN

35. Named after, and in homage to, the company that originally pioneered and developed CDN technologies in the 1990's, Sandpiper CDN now brings this suit to address Comcast's longstanding infringement of the patented technology claimed by the Asserted Patents.

36. The Asserted Patents are valid and enforceable, and the inventions claimed in the Asserted Patents are enabled, novel, non-obvious, unconventional, and non-routine at least as of their respective filing dates.

ASSERTED PATENTS

37. Patent Number 9,628,347 ("the '347 Patent") is entitled "Layered Request Processing in a Content Delivery Network (CDN)," and claims priority to U.S. Provisional Patent Application No. 61/737,072 filed on December 13, 2012. *See* Ex. A.

38. As CDNs became more widely adopted, the type of the content the networks were responsible for serving and delivering evolved. As a result, issues started to arise as CDNs—originally configured to treat any and all requests for content the same—started receiving different requests for different types of content, such as video, images, written work, or other multimedia content. Further compounding the issue were the varying needs of distinct origin servers across the CDN, each with their own unique configurations and properties.

39. The inventors of the '347 Patent pioneered technical solutions to these problems.

Instead of employing CDNs using a conventional one-size-fits all approach, the inventors developed the specific, technical solution of employing a layered approach for responding to requests. In doing so, certain embodiments of the '347 Patent configure layers for processing requests using a modifiable, modular control environment.[13]

40. The claims of the '347 patent describe specific, technical solutions, using specific structures, that were not conventional at the time of the invention of the '347 Patent, and the '347 Patent describes how the solutions are implemented. For example, independent claim 1 recites:

> processing said request, starting at said particular first layer, said processing being based on a modifiable runtime environment, said processing continuing conditionally through each of said particular layers in turn until either said request is terminated by one of said layers or said particular last layer processes said request . . . wherein, in processing of said request, at least one of said layers modifies said modifiable control environment to produce a modified control environment, and wherein processing of said request by a subsequent layer is based on the modified control environment.

The claimed solution provides numerous technical benefits specific to the technical field of CDNs that was unconventional at the time of filing the '347 Patent. First, the claimed "layered approach to request processing may provide for separate levels of configuration for each service."[14] Second, "the net effect[s of the '347 Patent] could be to augment or modify the subscriber/coserver sequence from what it might have been had the preceding layers not been executed."[15] Third, the operation of the hardware operating the CDN is improved in that the '347 Patent may "provide lower latency for the current request" and "improve performance or

[13] *See* '347 patent, 64:11-16.
[14] *See id.* at 67:37-38.
[15] *See id.* at 65:67-66:2.

availability of the service."[16]

41. Patent Number 9,660,876 (the '876 Patent) is entitled "Collector Mechanisms in a Content Delivery Network," and it claims priority to U.S. Provisional Patent Application No. 61/737,072 filed on December 13, 2012. *See* Ex. B.

42. As CDN technology developed, the type of the content served and delivered by CDNs evolved. As a result, CDNs—originally configured to treat requests for all content the same—experienced issues as they started receiving different requests for different types of content, such as video, images, written work, or other multimedia content. Further compounding the issue was the challenge in configuring CDNs to account for different types of parameters associated with delivering content, including responsibilities, security, quality, and the like.

43. The inventors of the '876 Patent identified these technical problems and invented specific and technical solutions to address them. In particular, the inventors developed, *inter alia*, a specific and technical solution whereby "state data are used to inform a peering policy of a set of caches," where a collector system "produce[s] state data relating to and based on information represented in said event data of said multiple event streams."[17] The inventors also discovered that using the state data relating to the event streams to inform a peering policy empowered the customization of computing components in a CDN, which improves computational efficiency of the CDN through specialization. Moreover, as the inventors explained to the Patent Office, the disclosed inventions of the '876 Patent are not abstract ideas, but are "applicable and limited by the claims to content delivery networks and provide[] a concrete way to support efficient and scalable content delivery in such networks . . . [and] '[are] necessarily rooted in computer

[16] *See id.* at 158:53-54 and 49:37.
[17] *See* '876 Patent, claim 1.

technology in order to overcome a problem specifically arising in the realm of computer networks.'"[18] The technical solution of the '876 Patent provides CDNs with the infrastructure and logic to handle specific requests for specific users in a more computationally efficient manner that was more scalable than other existing systems.[19]

44. Indeed, the claims of the '876 Patent include specific and technical solutions, using specific structure, that were not conventional at the time, and the '876 Patent describes how the solutions are implemented, as recited in the method of independent claim 1. First, using the claimed state data relating to the event streams to inform a peering policy allows for the customization of computing components in a CDN to improve computational efficiency of the CDN through specialization.[20] Second, using the claimed state data to inform a peering policy for a set of peer caches improves fault tolerance within a CDN to reduce outages.[21] Third, certain CDNs implementing embodiments claimed by the '876 Patent are able to account for the ordering of the event streams to improve efficiency in configuring the CDNs by receiving multiple event streams, each comprising a timestamp for the event.

---

[18] *See* Applicant Arguments/Remarks Made in an Amendment dated June 2, 2016, p. 12 (successfully traversing a Patent Office rejection under 35 U.S.C. § 101 to establish that the patent application for the '876 patent is directed to patent eligible subject matter) (citing *DDR Holdings, LLC v. Hotels.Com, L.P.*, 7773 F.3d 1245, 1257 (Fed. Cir. 2014)).

[19] *See id.* 36:29-32 ("This provides the benefits of command tracking for uncached resources in a more scalable way."); *see also* 39:38-40:28 (section covering "Efficiency of Group Handling").

[20] *See id.* at 52:16-22 ("The applicable chain of responsibilities and the capacity behind each are determined by the peering policy in effect based on [example state data]. This allows different request types to lead to different responsibility chains and different numbers of nodes allocated per responsibility.").

[21] *See id.* at 53:35-39 ("[W]hen a new responsibility allocation is provided (due to a node becoming completely unavailable or having its capacity metric degraded), the previous allocation and the new allocation are combined over some fade interval to determine the actual responsibility set used by any node.").

45. Patent Number 8,478,903 ("the '903 Patent") is entitled "Shared Content Delivery Infrastructure," and it claims priority to U.S. Patent Application No. 09/021,506, filed on February 10, 1998. *See* Ex. C.

46. As Internet use has increased, website owners must address ever-increasing bandwidth needs, dynamic changes in load, and performance issues relating to browsing clients, including clients in remote or distant locations.[22] When a server with website content, such as an origin server, receives multiple requests for website content, delivery of that content can be slow.

47. The invention of the '903 patent provides solutions to this problem. In certain embodiments of the '903 patent, methods are provided to, in a computer network, off-load processing of requests for selected resources (such as website content) by determining a different server to process those requests.[23] In some embodiments, this involves replicating content from a source associated with a client of a CDN network onto CDN servers. For example, end-user requests could be directed to the CDN servers instead of to the client's servers or the client's origin servers. These solutions in the '903 Patent improve the performance of providing website content to users in a network.[24]

48. Additionally, certain claimed embodiments of the '903 Patent solve issues related to delivering resources from more than one content provider. For instance, the inventors of the '903 Patent pioneered CDN technology for delivering resources associated with more than one content provider, embodiments of which involve a shared CDN server and alias names in order to provide resources in response to requests. In one example, an alias name is an alternate name that

[22] '903 Patent, 1:27-31.
[23] *Id*. at 2:62-65.
[24] *Id.*

can be used to make a connection. Certain claimed solutions of the '903 Patent include specific combinations that were not conventional at the time of the invention of the '903 Patent.

49. In one specific example of a technical improvement, a repeater server in a CDN maintains a partial mirror of more than one origin server by implementing a distributed and/or coherent cache of the origin server(s), in order to off-load processing and provide additional bandwidth for multiple website owners.[25] In one example, a repeater server is a network node that amplifies and rebroadcasts signals. In one example, a distributed cache is a type of cache that distributes data across multiple servers in a cluster, while a coherent cache ensures that all clients see the same data in a cache.

50. According to one claimed solution in the '903 Patent, a content delivery system with at least one shared repeater server is provided. The shared repeater server can replicate resources associated with origin servers. In one embodiment, requests for a resource on an origin server are directed, at least in part, based on an alias name to at least one repeater server.[26]

51. Patent Number 7,013,322 (the '322 Patent) is entitled "System and Method for Rewriting a Media Resource Request and/or Response between Origin Server and Client," and claims priority to U.S. Provisional Patent Application 60/178,750 filed on January 28, 2000. *See* Ex. D.

52. At the time of the inventions disclosed in the '322 Patent, the Internet was becoming a widely used medium for communicating and distributing information. As the '322 Patent explains, "[s]ince the Internet is essentially a network of connected computers distributed throughout the world, the activity performed by each computer or server to transfer information

[25] *Id*. at 4:37-40.
[26] *Id*. at claim 28.

from a particular source to a particular destination naturally increases in conjunction with increased Internet use."[27]

53. In one embodiment, each computer is referred to us as a "node." The transfer of data from one computer or node to another is commonly referred to as a "hop." At the time of the invention, the inventors of the '322 Patent noticed the huge volume of data that each computer or node was transferring on a daily basis. The inventors of the '322 Patent determined that minimizing the amount of hops taken to transfer data from a source to a particular destination or end user would minimize the amount of computers or nodes needed for a data transfer.

54. The claimed embodiments of the '322 Patent address these problems. The '322 Patent proposes a revolutionary computer network capable of modifying media resource request metafiles, as well as the responses to those media resource requests by media servers in the network, to provide more efficient content delivery in the network.

55. The claims of the '322 Patent include specific solutions, using specific structure, that were not conventional at the time of the invention of the '322 Patent, and the '322 Patent describes how the solutions are implemented. For example, independent claim 11 recites, *inter alia*, "intercepting and modifying a data resource request issued by a user requesting data from said network," and independent claim 16 recites, *inter alia*, "intercepting and modifying a data response issued by a data server in said network in response to a data resource request issued by a user."

56. These claimed solutions provide numerous technical benefits specific to the technical field of CDNs and were unconventional at the time the invention of the '322 Patent. First, "intercepting a media resource request metafile or a response to the media resource request

[27] '322 patent at 2:5-10.

by a media server in the network, and intelligently re-writing" the request or response "improve[s] content delivery capability of the network."[28] Second, the "distributed network therefore need not have a centralized structure where everything about the network is known at one location. Instead, the central location can send back a response that it deems most appropriate, and other networks along the path of travel of the response can likewise intercept and change this request to 'fine tune' the response for their respective network."[29]

57. Patent Number 9,762,692 ("the '692 Patent") is entitled "Handling long-tail content in a content delivery network (CDN)." The '692 Patent claims priority to U.S. Provisional Patent Application No. 61/042,412, filed on April 4, 2008. *See* Ex. E.

58. CDNs generally provide multiple servers to serve content. Issues can arise in CDNs when responding to requests for content, such as technical issues relating to handling requests and timely providing content. For example, certain content can become popular or fade into obscurity dynamically over time.[30] CDNs face technical issues with respect to serving content that is popular, or not popular, in a network with multiple servers.[31] Moreover, CDNs can experience increased latency and degraded service if content is cached at a particular server even if the content is not popular.[32] The risk of "thrashing" to serve content to all customers is also an issue for CDNs when content is unnecessarily cached.[33]

59. The inventors of the '692 Patent developed specific, technical solutions to these problems, which are described in the '692 Patent. The '692 Patent describes infrastructure and

[28] '322 patent at 3:45-50.
[29] *Id*. at 3:62-4:4.
[30] '692 patent, 2:46-51.
[31] *Id*.
[32] *Id*. at 10:1-22.
[33] *Id*.

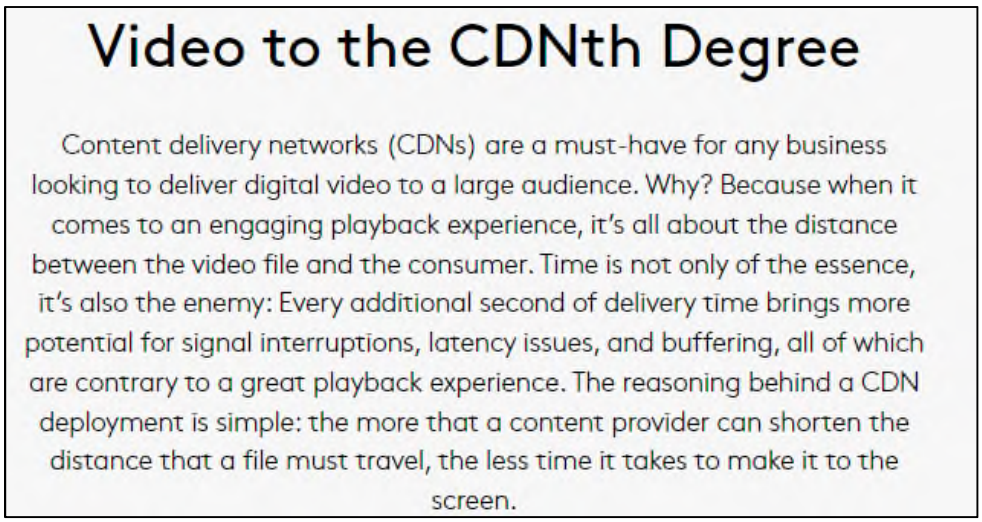
logic for delivering content in a content delivery network including a first tier of servers. In some embodiments, a server in the first tier of servers obtains a request for a resource that is available as part of a content provider's library. If the resource is not available at the server or a peer of the server, certain embodiments determine if the resource is popular. If the resource is determined to be popular, the server can obtain and serve the resource. If the resource is not determined to be popular, the requester can be directed to a second server in a second tier of servers, which serves the resource, for example. In some embodiments, a portion of the content provider's library comprising the second server is distinct from a portion on another server in the second tier.

60. The '692 Patent describes a CDN including a tiered server system. Certain embodiments claimed by the '692 Patent address the reality of storage limitations by setting forth a framework that obtains only popular content at a first tier of servers, thus both speeding up content delivery and preserving memory space for popular content in a first tier. The inventors of the '692 Patent pioneered specific, technical solutions for using tiers of servers and specific processes for serving resources in a memory conscious manner that addresses latency issues. Certain claimed embodiments of the '692 Patent include particular implementations and combinations that were not conventional at the time of the invention of the '692 Patent, providing specific technical improvements to CDNs.

THE ACCUSED COMCAST CDN FUNCTIONALITIES

61. The accused Comcast CDN functionalities comprise the Comcast products and services using Apache Traffic Control and any of Comcast's bespoke CDN architecture or solutions. For example, Comcast uses the accused Comcast CDN functionalities for its own products and services, and sells CDN services to businesses and other content providers via its Content Delivery Network (CDN) Suite. Comcast's motivation for infringing and continuing to

infringe the Asserted patents is due at least in part to its growing demand for delivering content to customers, the infrastructure of which is built based on CDN technology claimed by the Asserted Patents as provided below.



https://www.comcasttechnologysolutionscom/content-delivery-network-cdn-suite/content-delivery

COUNT I: INFRINGEMENT OF THE '347 PATENT

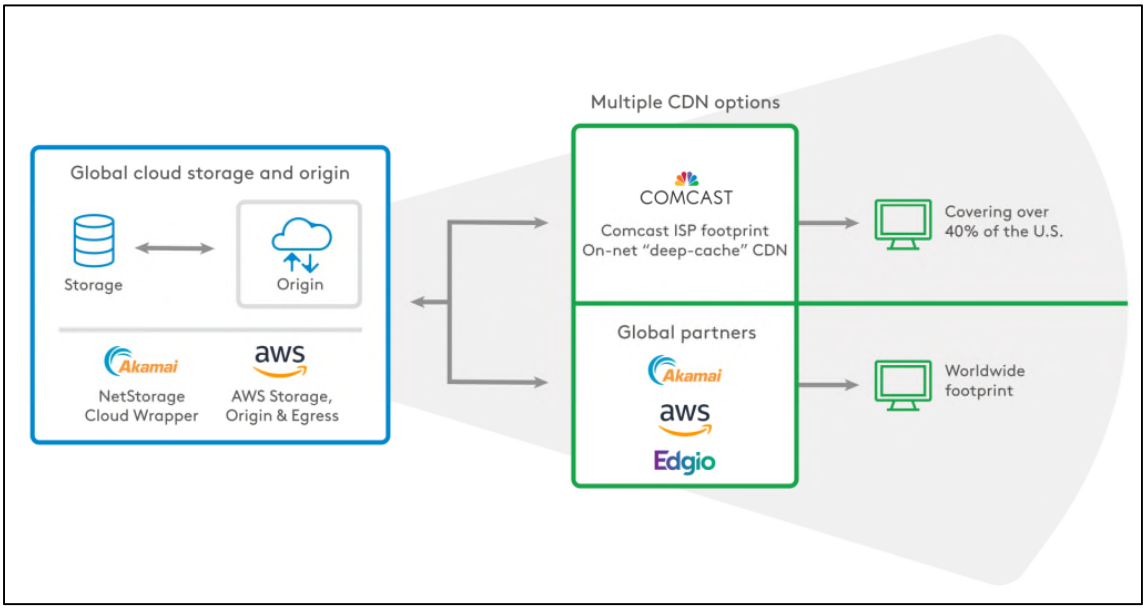
62. Plaintiff hereby incorporates by reference each of the allegations in the foregoing paragraphs as though fully set forth herein, and further alleges as follows.

63. Taking advantage of the technical improvements described in the '347 Patent, the accused Comcast CDN functionalities have, pursuant to 35 U.S.C. § 271, infringed and continue to infringe the claimed layered approach for configuring the accused Comcast CDN functionalities to improve content delivery. Through at least its version of Traffic Control, the accused Comcast CDN functionalities perform a computer-implemented method in a content delivery network (CDN) that infringes at least claims 1, 2, and 4 of the '347 Patent.[34] Comcast's infringement has

[34] Although Sandpiper cites to claims 1, 2, and 4, Sandpiper does not concede that any one claim is representative of the entire '347 Patent nor does Sandpiper submit that Comcast is only infringing these claims. Rather, Sandpiper believes that claims in the '347 Patent are patentably distinct and the claims of the '347 Patent support alternative or additional infringement theories.

caused and will continue to cause damages for which Plaintiff is entitled to compensation pursuant to 35 U.S.C. § 284.

64. As a non-limiting example, claim 1 of the '347 Patent generally recites a computer-implemented method in a content delivery network on a device comprising hardware including memory and at least one processor. On information and belief, the accused Comcast CDN functionalities offer its CDN functionalities across its CDN utilizing distributed system of servers that comprise memory and processors:[35]



1.1.1 Content Delivery Networks

The vast majority of today's Internet traffic is media files (often video or audio) being sent from a single source (the *Content Provider*) to many thousands or even millions of destinations (the *Content Consumers*). CDN (Content Delivery Network)s are the technology that make that one-to-many distribution efficient. A CDN is a distributed system of servers for delivering content over HTTP(S). These servers are deployed in multiple locations with the goal of optimizing the delivery of content to the end users, while minimizing the traffic on the network. A CDN typically consists of the following:

65. The first step of the method in claim 1 generally recites receiving a request for a

[35] https://www.comcasttechnologysolutions.com/content-delivery-network-cdn-solutions; https://readthedocs.org/projects/traffic-control-cdn/downloads/pdf/latest/.

content delivery service of a particular service type. On information and belief, the accused Comcast CDN functionalities perform this step by utilizing ATC to allow a traffic router or traffic server to receive a client request for a particular type of service. After doing so, the accused Comcast CDN functionalities perform a sequence of events upon receiving the request for a service of a particular service type:[36]

#21 Example Client Request to Traffic Router

```
GET / HTTP/1.1
Host: video.demo1.mycdn.ciab.test
Accept: */*
```

What follows is a sequence of events that take place when a client requests content from an HTTP/1.1-compliant server.

1. The client sends a request to the LDNS (Local DNS) server to resolve the name `www.origin.com` to an IP address, then sends an HTTP request to that IP.

Instead, Traffic Server provides special event-driven mechanisms for efficiently scheduling work: the event system and continuations. The **event system** is used to schedule work to be done on threads. A **continuation** is a passive, event-driven state machine that can do some work until it reaches a waiting point; it then sleeps until it receives notification that conditions are right for doing more work. For example, HTTP state machines (which handle HTTP transactions) are implemented as continuations.

66. The first step of the method in claim 1 also generally recites that the content delivery service defines a particular number of configurable layers of request processing associated with the content delivery service and particular service type in sequential order from a particular first layer to a particular last layer. On information and belief, the accused Comcast CDN

[36] https://traffic-control-cdn.readthedocs.io/en/latest/overview/traffic_router.html#traffic-router; https://readthedocs.org/projects/traffic-control-cdn/downloads/pdf/latest/; https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

functionalities do so when the traffic servers utilized in the accused Comcast CDN functionalities provide a continuation for handling Hypertext Transfer Protocol (HTTP) transactions:[37]

> Instead, Traffic Server provides special event-driven mechanisms for efficiently scheduling work: the event system and continuations. The **event system** is used to schedule work to be done on threads. A **continuation** is a passive, event-driven state machine that can do some work until it reaches a waiting point; it then sleeps until it receives notification that conditions are right for doing more work. For example, HTTP state machines (which handle HTTP transactions) are implemented as continuations.

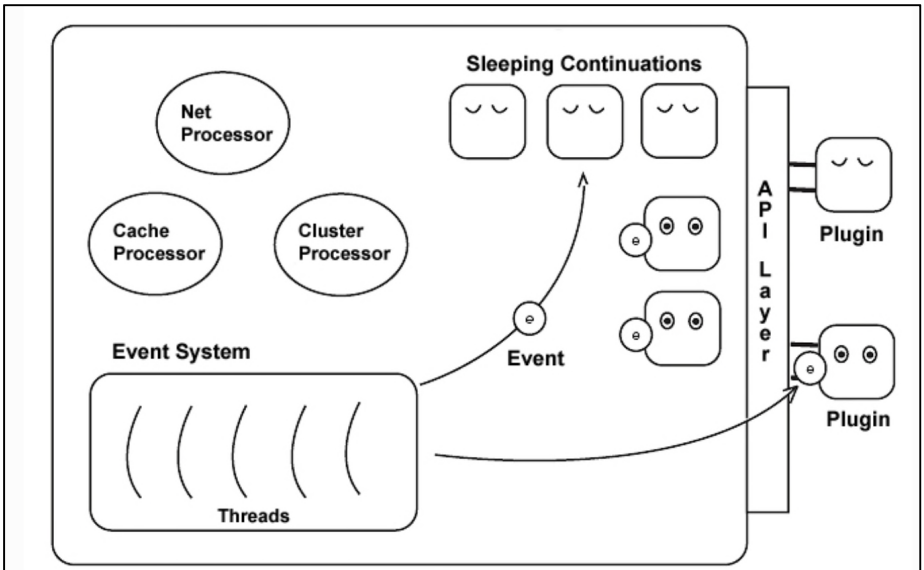
> Continuation objects are used throughout Traffic Server. Some might live for the duration of the Traffic Server process, while others are created (perhaps by other continuations) for specific needs and then destroyed. Traffic Server Internals (below) shows how the major components of Traffic Server interact. Traffic Server has several **processors**, such as *cache processor* and *net processor*, that consolidate cache or network I/O tasks. Processors talk to the event system and schedule work on threads. An executing thread calls back a continuation by sending it an event. When a continuation receives an event, it wakes up, does some work, and either destroys itself or goes back to sleep & waits for the next event.

> Plugins are typically implemented as continuations. All of the sample code plugins (except `hello_world`) are continuations that are created when Traffic Server starts up; they then wait for events that trigger them into activity.

67. The second step of the method of claim 1 generally recites processing the request, starting in the particular first layer, where the processing is based on a modifiable runtime environment and continues conditionally thorough each of the particular layers until the request is terminated by one of the particular layers or the particular last layer processes the request. On information and belief, ATC, utilized by the accused Comcast CDN functionalities, supports a variety of plugins that can be sequentially connected to form configurable layers of request processing sequentially from a first layer to a last layer. Example plugins used by the accused Comcast CDN functionalities include the Denylist Plugin, the Compress Plugin, the Header

[37] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

Manipulation Plugins, and others.[38] On information and belief, the accused Comcast CDN functionalities deploy these plugins or similar software to process the request, starting at the particular first layer and based on a modifiable runtime environment. This processing continues conditionally through each of the particular layers in turn until either the request is terminated by one of said layers or the particular last layer processes the request:[39]



Traffic Server with Plugins

The example above shows a simple plugin that creates a continuation and then schedules it to be called immediately. When the plugin's handler function is called the first time, the event is `TS_EVENT_IMMEDIATE`. The plugin then tries to open a net connection to port 9999 on `localhost` (127.0.0.1). The IP description was left in cider notation to further clarify what is going on; also note that the above won't actually compile until the IP address is modified. The action returned from `TSNetConnect` is examined by the plugin. If the operation has not completed, then the plugin stores the action in its continuation. Otherwise, the plugin knows it has already been called back and there is no reason to store the action pointer.

[38] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

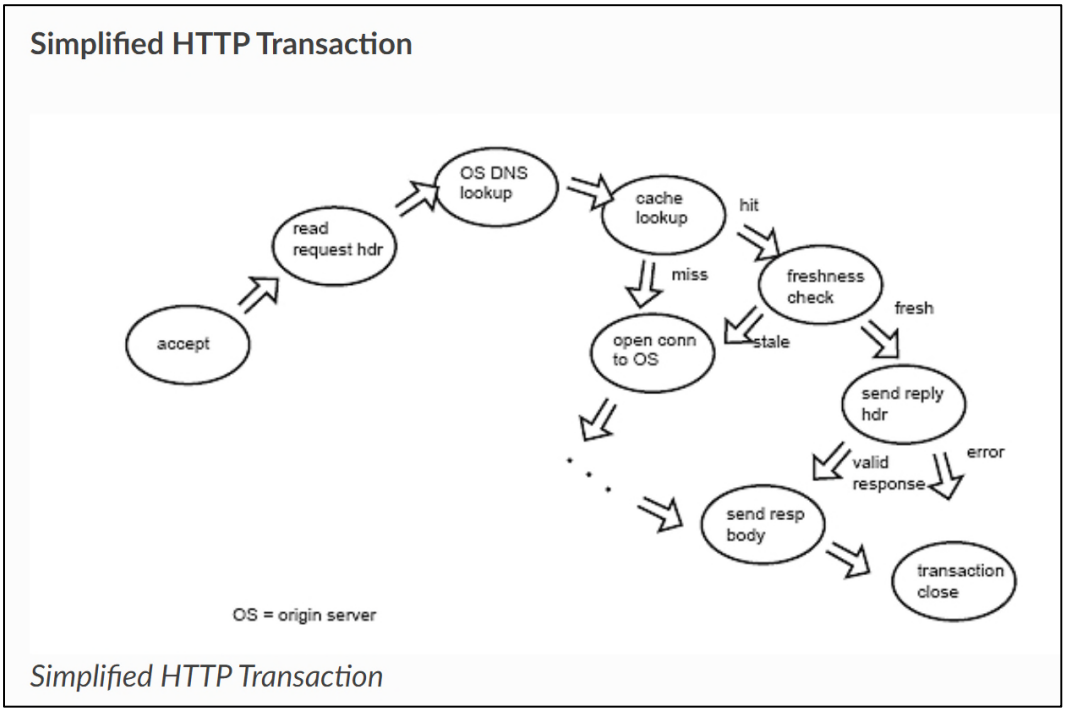
[39] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html; https://docs.trafficserver.apache.org/developer-guide/plugins/actions/index.en.html; https://docs.trafficserver.apache.org/admin-guide/plugins/http_stats.en.html; https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

> A plugin may consist of just one static continuation that is called whenever certain events happen. Examples of such plugins include `denylist_1.cc`, `basic_auth.cc`, and `redirect_1.cc`. Alternatively, a plugin might dynamically create other continuations as needed. Transform plugins are built in this manner: a static parent continuation checks all transactions to see if any are transformable; when a transaction is transformable, the static continuation creates a type of continuation called a **vconnection**. The vconnection lives as long as it takes to complete the transform and then destroys itself. This design can be seen in all of the sample transform plugins. Plugins that support new protocols also have this architecture: a static continuation listens for incoming client connections and then creates transaction state machines to handle each protocol transaction.

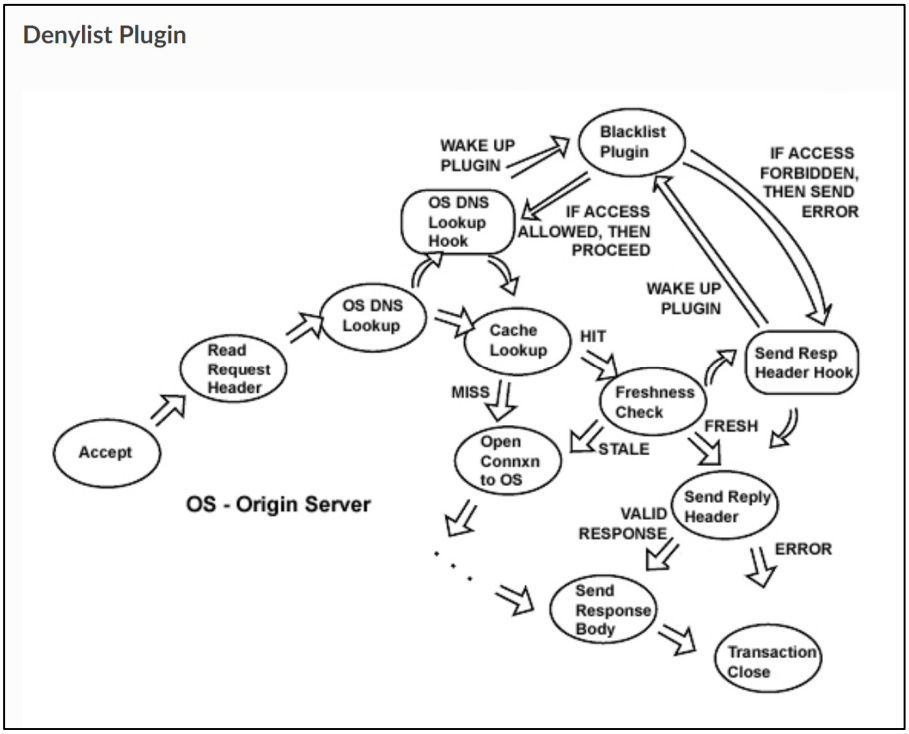
68. The runtime environment of claim 1 comprises a modifiable control environment and a modifiable request environment. Upon information and belief, the accused Comcast CDN functionalities include a modifiable control environment which allows servers to be configured and assigned to respective services. The accused Comcast CDN functionalities are also configurable to change and provide additional functionalities via various plugins. For example, on information and belief, the accused Comcast CDN functionalities implement, via ATC, a Denylist plugin to modify performing an HTTP transaction or other response.[40]

> **Servers**
>
> Servers can be assigned to Delivery Services using the Servers and Delivery Services Traffic Portal sections, or by directly using the deliveryserviceserver endpoint. Only Edge-tier cache servers can be assigned to a Delivery Service, and once they are so assigned they will begin to serve content for the Delivery Service (after updates are queued and then applied). Any servers assigned to a Delivery Service must also belong to the same CDN as the Delivery Service itself. At least one server must be assigned to a Delivery Service in order for it to serve any content.

[40] https://traffic-control-cdn.readthedocs.io/en/latest/overview/delivery_services.html#type; https://traffic-control-cdn.readthedocs.io/en/latest/overview/delivery_services.html#type.



Simplified HTTP Transaction



> The Denylist plugin's hook to the origin server DNS lookup state is a *global hook*, meaning that the plugin is called every time there's an HTTP transaction with a DNS lookup event. The plugin's hook to the send reply header state is a *transaction hook*, meaning that this hook is only invoked for specified transactions (in the Denylist Plugin example, it's only used for requests to denylisted servers). Several examples of setting up hooks are provided in Header-Based Plugin Examples and HTTP Transformations.

69. The modifiable request environment of claim 1 is also based on the request. On information and belief, the accused Comcast's CDN functionalities provide a Compress plugin, which is used to configure the modifiable request environment to respond with compressed or uncompressed content based on the request received.[41]

> **Compress Plugin** ¶
>
> This plugin adds compression and decompression options to both origin and cache responses.

> Not all clients can handle compressed content. Not all origin servers are configured to respond with compressed content when a client says it can accept it. And it's not always necessary to make two separate requests to an origin, and track two separate cache objects, for the same content - once for a compressed version and another time for an uncompressed version.
>
> This plugin tidies up these problems by transparently compressing or deflating origin responses, as necessary, so that both variants of a response are stored as alternates and the appropriate version is used for client responses, depending on the client's indication (via an `Accept` request header) of what it can support.

As another example, on information and belief, the accused Comcast CDN functionalities provide plugins that modify the modifiable request environment based on the request when execution of the plugin may dynamically create other continuations or build other plugins as needed to process

[41] https://docs.trafficserver.apache.org/admin-guide/plugins/compress.en.html.

a particular request.[42]

> A plugin may consist of just one static continuation that is called whenever certain events happen. Examples of such plugins include `denylist_1.cc`, `basic_auth.cc`, and `redirect_1.cc`. Alternatively, a plugin might dynamically create other continuations as needed. Transform plugins are built in this manner: a static parent continuation checks all transactions to see if any are transformable; when a transaction is transformable, the static continuation creates a type of continuation called a **vconnection**. The vconnection lives as long as it takes to complete the transform and then destroys itself. This design can be seen in all of the sample transform plugins. Plugins that support new protocols also have this architecture: a static continuation listens for incoming client connections and then creates transaction state machines to handle each protocol transaction.

70. The modifiable control environment of claim 1 also comprises a modifiable global control environment that is distinct from the modifiable request environment. On information and belief, in one implementation, the accused Comcast CDN functionalities deploy a header manipulation plugin or similar software that includes performs authorization or redirects using hooks. Further, on information and belief, the modifiable control environment in the accused Comcast CDN functionalities is distinct from the modifiable request environment:[43]

> *Header manipulation plugins*, such as filtering, basic authorization, or redirects, usually have a global hook to the DNS lookup or the read request header states. If specific actions need to be done to the transaction further on, then the plugin adds itself to a transaction hook. *Transformation plugins* require a global hook to check all transactions for transformability followed by a *transform hook*, which is a type of transaction hook used specifically for transforms.

[42] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

[43] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html; https://traffic-control-cdn.readthedocs.io/en/latest/overview/delivery_services.html#type.

> **Servers**
>
> Servers can be assigned to Delivery Services using the Servers and Delivery Services Traffic Portal sections, or by directly using the deliveryserviceserver endpoint. Only Edge-tier cache servers can be assigned to a Delivery Service, and once they are so assigned they will begin to serve content for the Delivery Service (after updates are queued and then applied). Any servers assigned to a Delivery Service must also belong to the same CDN as the Delivery Service itself. At least one server must be assigned to a Delivery Service in order for it to serve any content.

71. In addition, for the request initially received in step one of the method of claim 1, the first layer determines the modifiable request environment from information associated with that request, and then each subsequent layer obtains the modifiable request environment from the previous layer. On information and belief, the accused Comcast CDN functionalities provide hooks to determine the modifiable environment, initiates the plugin, and performs the associated functionalities:[44]

> **Use Case 1: Proxy only mode using HTTP cookies.**
>
> **<01-02>**. When a request from the UA is received at the CDN the value of `TokenCookie` is extracted and the access token is validated (missing cookie or access token is same as invalid).
>
> **<03>**. If the access token is valid its opaque *subject* is extracted, added to the cache key and a cache lookup is performed.
>
> **<06>**. Missing or invalid access token or a cache-miss leads to forwarding the request to the origin either for user authorization and/or for fetching the object from origin.

[44] https://docs.trafficserver.apache.org/en/latest/admin-guide/plugins/access_control.en.html; *see also* https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

> You use hooks as triggers to start your plugin. The name of a hook reflects the Traffic Server state that was *just completed*. For example, the "OS DNS lookup" hook wakes up a plugin right *after* the origin server DNS lookup. For a plugin that requires the IP address of the requested origin server, this hook is the right one to use. The Denylist plugin works in this manner, as shown in the Denylist Plugin diagram below.

On information and belief, the accused Comcast CDN functionalities ensure that each subsequent layer obtains the modifiable request environment from the previous layer by, for example, leveraging ATC to provide continuation through subsequent layers and deploying plugins or other software that can be linked to libraries to transmit information there between.[45]

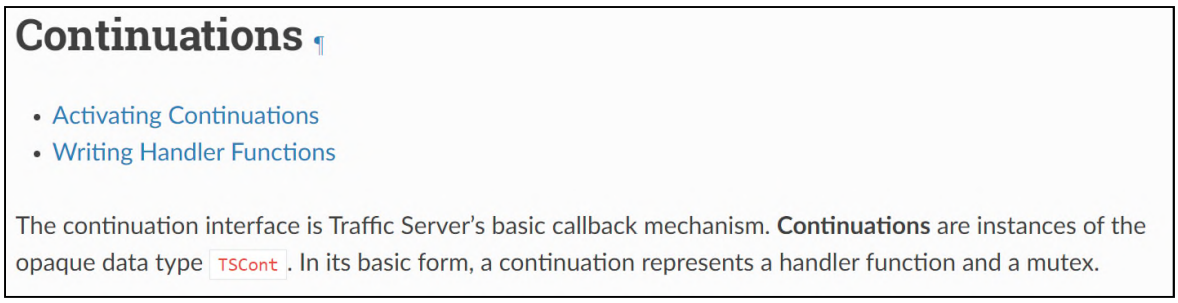
> **Continuations** ¶
>
> - Activating Continuations
> - Writing Handler Functions
>
> The continuation interface is Traffic Server's basic callback mechanism. **Continuations** are instances of the opaque data type `TSCont`. In its basic form, a continuation represents a handler function and a mutex.

> This example `CMakeLists.txt` finds the tsapi package which provides the `add_atsplugin` and `verify_remap_plugin` functions. `add_atsplugin` does all of the necessary cmake commands to build a plugins module .so. The function takes the plugin target name and a list of source files that make up the project. If the plugin requires additional libraries, those can be linked with the `target_link_libraries` cmake function. If your plugin links OpenSSL (or the alternatives), put it in the beginning part of the list to avoid link issues.

72. Claim 1 also generally recites, in processing the request received in step 1 of the method, that at least one of the layers modifies the modifiable control environment to produce a modified control environment. On information and belief, the accused Comcast CDN

[45] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/continuations/index.en.html#developer-plugins-continuations; *see also* https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/building-plugins.en.html; *see also* https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

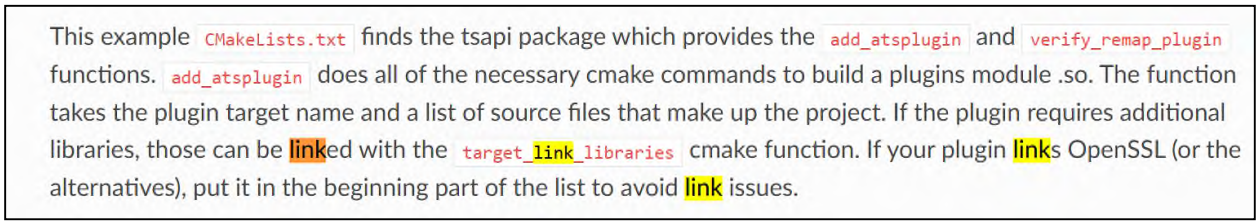
functionalities deploy plugins or other software that can be linked to libraries to transmit information there between. For example, certain plugins employed by the accused Comcast CDN functionalities, such as the remap plugin, can be chained to affect the output.[46]



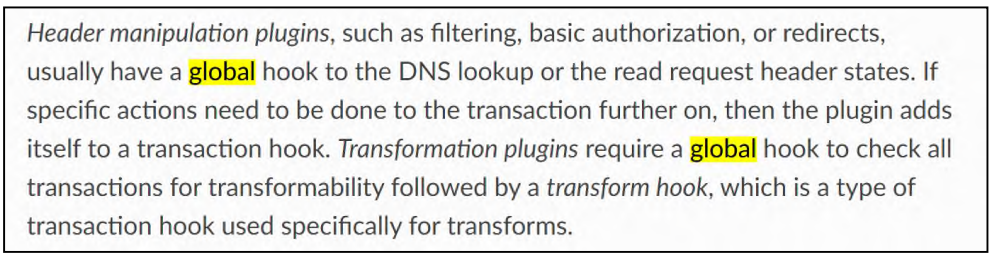
Continuations ¶

- Activating Continuations
- Writing Handler Functions

The continuation interface is Traffic Server's basic callback mechanism. **Continuations** are instances of the opaque data type `TSCont`. In its basic form, a continuation represents a handler function and a mutex.



This example `CMakeLists.txt` finds the tsapi package which provides the `add_atsplugin` and `verify_remap_plugin` functions. `add_atsplugin` does all of the necessary cmake commands to build a plugins module .so. The function takes the plugin target name and a list of source files that make up the project. If the plugin requires additional libraries, those can be linked with the `target_link_libraries` cmake function. If your plugin links OpenSSL (or the alternatives), put it in the beginning part of the list to avoid link issues.



Header manipulation plugins, such as filtering, basic authorization, or redirects, usually have a global hook to the DNS lookup or the read request header states. If specific actions need to be done to the transaction further on, then the plugin adds itself to a transaction hook. *Transformation plugins* require a global hook to check all transactions for transformability followed by a *transform hook*, which is a type of transaction hook used specifically for transforms.

Claim 1 further claims this processing of the request by a subsequent layer based on the modified control environment and where at least one of the subsequent layers modifies the runtime environment. The accused Comcast CDN functionalities process requests in at least one layer to modify said runtime environment, for example, where a remap plugin is reloaded during runtime to modify the runtime environment as part of processing the request:[47]

[46] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/continuations/index.en.html#developer-plugins-continuations; https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/building-plugins.en.html; https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

[47] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html; https://docs.trafficserver.apache.org/developer-guide/plugins/remap-plugins.en.html#runtime.

> *Header manipulation plugins*, such as filtering, basic authorization, or redirects, usually have a global hook to the DNS lookup or the read request header states. If specific actions need to be done to the transaction further on, then the plugin adds itself to a transaction hook. *Transformation plugins* require a global hook to check all transactions for transformability followed by a *transform hook*, which is a type of transaction hook used specifically for transforms.

> In addition, as of Traffic Server 9.0, remap plugins can be reloaded during runtime. During a `remap.config` reload, if the plugin image file has changed, a new one is loaded and used.
>
> All of the externally invoked functions must be declared as `extern "C"` in order to be correctly located by the Traffic Server core. This is already done if include/ts/remap.h is included, otherwise it must be done explicitly.

73. Claim 2 of the ’347 Patent depends from claim 1, and claims the method of claim 1 where, in processing the request, at least one of the layers modifies the request environment to produce a modified request environment. On information and belief, the accused Comcast CDN functionalities, in processing a request, causes at least one of said layers to modify the request environment to produce a modified request environment by, for example, employing the remap plugin, the header manipulation plugin, or similar software to perform operations that produce a modified request environment, and these plugins or similar software can be linked.[48]

> In addition, as of Traffic Server 9.0, remap plugins can be reloaded during runtime. During a `remap.config` reload, if the plugin image file has changed, a new one is loaded and used.
>
> All of the externally invoked functions must be declared as `extern "C"` in order to be correctly located by the Traffic Server core. This is already done if include/ts/remap.h is included, otherwise it must be done explicitly.

[48] https://docs.trafficserver.apache.org/developer-guide/plugins/remap-plugins.en.html#runtime; *see also* https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.

> *Header manipulation plugins*, such as filtering, basic authorization, or redirects, usually have a global hook to the DNS lookup or the read request header states. If specific actions need to be done to the transaction further on, then the plugin adds itself to a transaction hook. *Transformation plugins* require a global hook to check all transactions for transformability followed by a *transform hook*, which is a type of transaction hook used specifically for transforms.

Claim 2 further claims that processing of the request by a subsequent layer is based on the modified request environment. The accused Comcast CDN functionalities processes the request accordingly by, for example, utilizing ATC to cause a request to be processed by loading a new file and using the new file based on a reload executed by the reload plugin:[49]

> In addition, as of Traffic Server 9.0, remap plugins can be reloaded during runtime. During a `remap.config` reload, if the plugin image file has changed, a new one is loaded and used.
>
> All of the externally invoked functions must be declared as `extern "C"` in order to be correctly located by the Traffic Server core. This is already done if include/ts/remap.h is included, otherwise it must be done explicitly.

74. Claim 4 of the '347 Patent depends from claim 1, and claims the method of claim 1, where each layer determines how to process the request based on information in the runtime environment at the time the request reaches the each layer. On information and belief, the accused Comcast CDN functionalities determine how to process the request based on information in the runtime environment at the time the request reaches the layer, for example, by implementing rules for invoking a plugin using layers.[50]

[49] https://docs.trafficserver.apache.org/developer-guide/plugins/remap-plugins.en.html#runtime.
[50] https://docs.trafficserver.apache.org/developer-guide/plugins/remap-plugins.en.html#runtime.

> In addition, as of Traffic Server 9.0, remap plugins can be reloaded during runtime. During a `remap.config` reload, if the plugin image file has changed, a new one is loaded and used.
>
> All of the externally invoked functions must be declared as `extern "C"` in order to be correctly located by the Traffic Server core. This is already done if include/ts/remap.h is included, otherwise it must be done explicitly.

> In essence, `TSRemapNewInstance()` is called to create an invocation instance for the plugin to store rule local data. If the plugin is invoked multiples time on a rule, this will be called multiple times for the rule, once for each invocation. Only the value store in *ih* will be available when the rule is actually matched. In particular the plugin arguments will not be available.

COUNT II: INFRINGEMENT OF THE '876 PATENT

75. Plaintiff hereby incorporates by reference each of the allegations in the foregoing paragraphs as though fully set forth herein, and further alleges as follows.

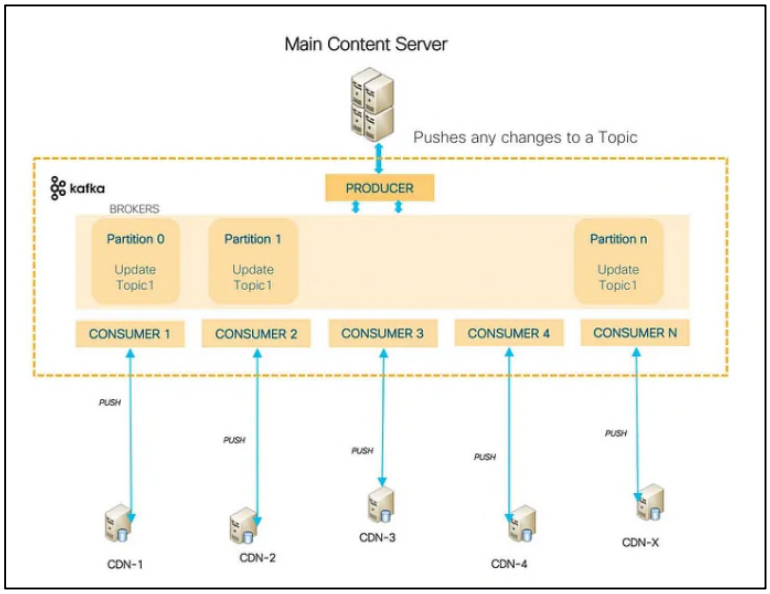
76. Taking advantage of the technical improvements described in the '876 Patent, the accused Comcast CDN functionalities have, pursuant to 35 U.S.C. § 271, infringed and continue to infringe the infringe the '876 Patent. For example, upon information and belief, the accused Comcast CDN functionalities employ a version of Kafka or Apache Kafka to perform a computer-implemented method in Comcast's CDN that infringes at least claim 1 of the '876 Patent. Comcast's infringement has caused and will continue to cause damages for which Plaintiff is entitled to compensation pursuant to 35 U.S.C. § 284.

77. As a non-limiting example,[51] claim 1 of the '876 Patent claims a computer-implemented method in a CDN including multiple content delivery services and operable on at

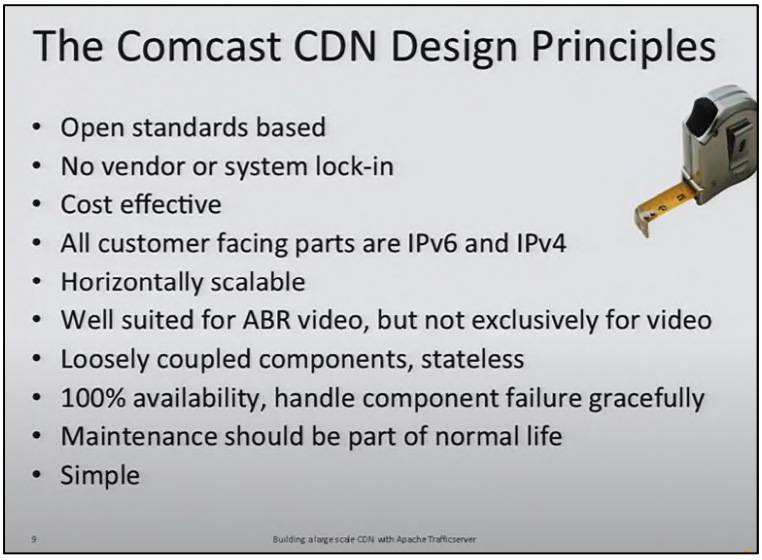
[51] Although Sandpiper cites to only claim 1, Sandpiper does not concede that any claim is representative of the entire '876 Patent nor does Sandpiper submit that Comcast is only infringing this claim. Rather, Sandpiper believes that claims in the '876 Patent are patentably distinct, and the claims of the '876 Patent support alternative or additional infringement theories.

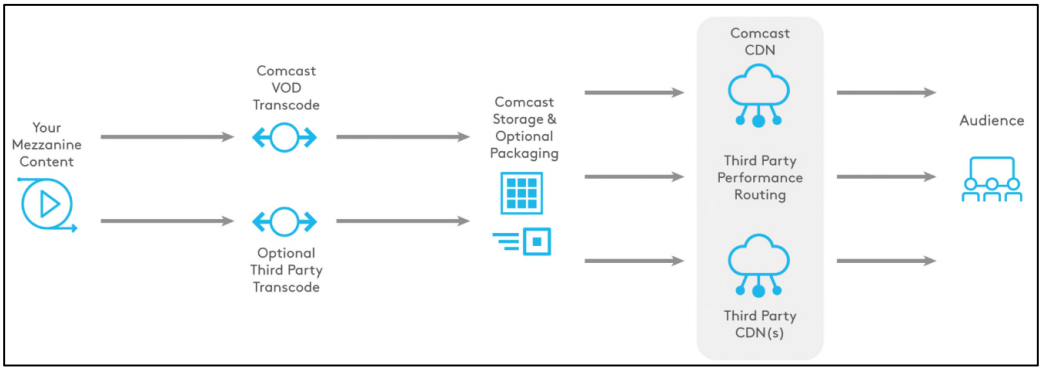
least one device with hardware including memory and at least one processor. The accused Comcast CDN functionalities implement the method of claim 1 to provide content delivery services utilizing on a hardware device with at least memory and a processor. For example, on information and belief, Apache Kafka supports "event streaming . . . applied to a variety of use cases across a plethora of industries and organizations," including event streaming within the accused Comcast CDN technologies and related infrastructure.[52]

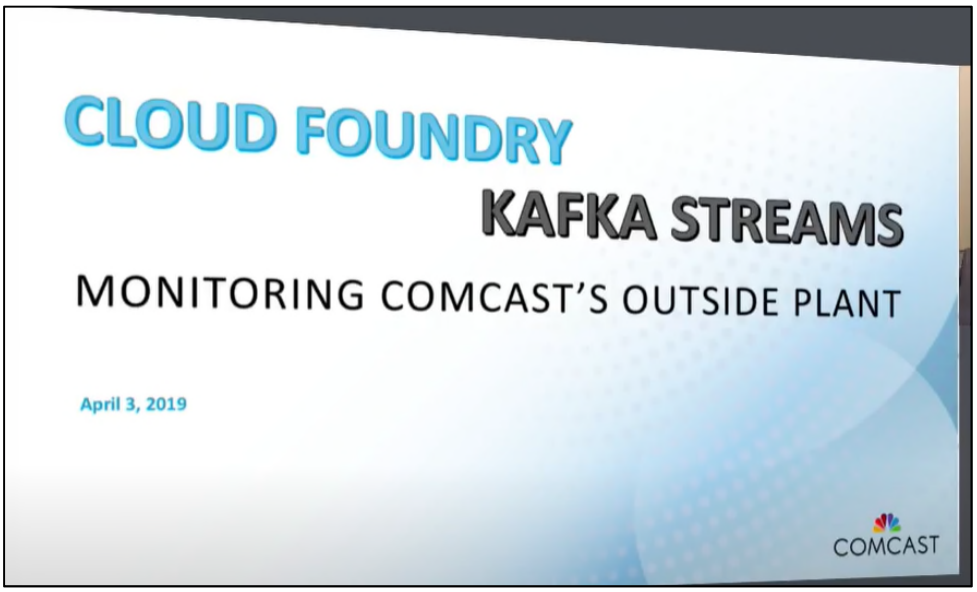
Architecture: Efficient way for updating a CDN Cluster using Kafka



[52] https://kafka.apache.org/documentation/; https://medium.com/@hafandhalf/architecture-efficient-way-for-updating-a-cdn-cluster-using-kafka-b2e7fa36a322; *see also* https://www.youtube.com/watch?v=q1mndAYZlio ("Building a large scale CDN with Apache Traffic Server - Jan van Doorn," posted April 25, 2014, by ApacheCon North America 2014); *see also* https://www.comcasttechnologysolutions.com/content-delivery-network-cdn-suite/content-delivery; *see also* https://www.youtube.com/watch?v=oVvoqg-NiKU ("Deep Dive: Cloud Foundry Stream Processing – Monitoring Comcast's Outside Plant - Charles (Mike) Graham & Dan Carroll, Comcast Cable Communications," posted April 15, 2019 by Cloud Foundry).







78. The first step of the method in claim 1 generally recites receiving multiple event

streams of event data. On information and belief, the accused Comcast CDN functionalities receive multiple event streams by running a cluster of one or more servers organized as a collector system capable of receiving multiple event streams:[53]

Apache Kafka® is an event streaming platform. What does that mean?

Kafka combines three key capabilities so you can implement your use cases for event streaming end-to-end with a single battle-tested solution:

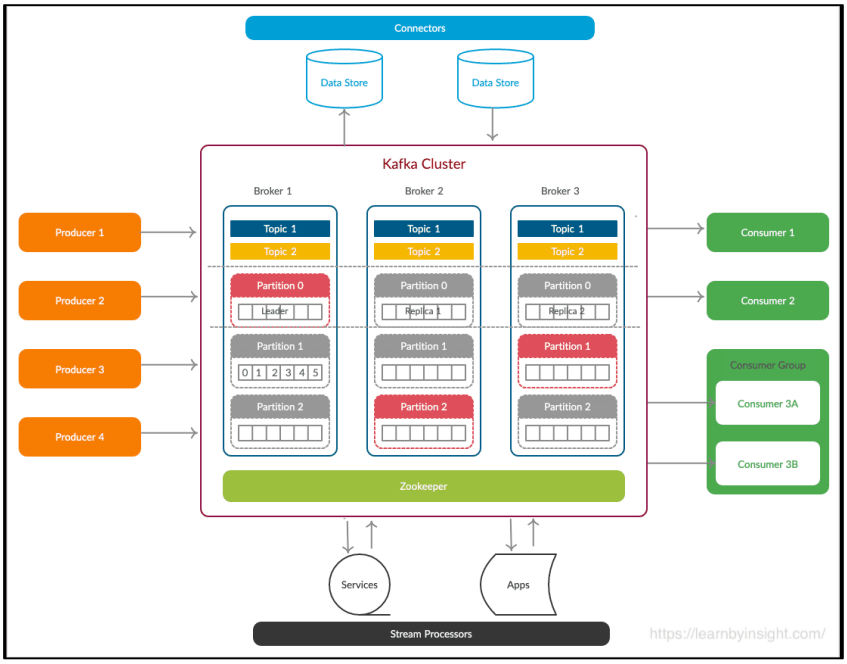
1. To **publish** (write) and **subscribe to** (read) streams of events, including continuous import/export of your data from other systems.
2. To **store** streams of events durably and reliably for as long as you want.
3. To **process** streams of events as they occur or retrospectively.

And all this functionality is provided in a distributed, highly scalable, elastic, fault-tolerant, and secure manner. Kafka can be deployed on bare-metal hardware, virtual machines, and containers, and on-premises as well as in the cloud. You can choose between self-managing your Kafka environments and using fully managed services offered by a variety of vendors.

Servers: Kafka is run as a cluster of one or more servers that can span multiple datacenters or cloud regions. Some of these servers form the storage layer, called the brokers. Other servers run Kafka Connect to continuously import and export data as event streams to integrate Kafka with your existing systems such as relational databases as well as other Kafka clusters. To let you implement mission-critical use cases, a Kafka cluster is highly scalable and fault-tolerant: if any of its servers fails, the other servers will take over their work to ensure continuous operations without any data loss.

Clients: They allow you to write distributed applications and microservices that read, write, and process streams of events in parallel, at scale, and in a fault-tolerant manner even in the case of network problems or machine failures. Kafka ships with some such clients included, which are augmented by dozens of clients provided by the Kafka community: clients are available for Java and Scala including the higher-level Kafka Streams library, for Go, Python, C/C++, and many other programming languages as well as REST APIs.

[53] https://kafka.apache.org/documentation/; *see also* https://avesha.io/resources/blog/kafka-multi-cluster-deployment-on-kubernetes-simplified.



79. The first step of the method in claim 1 also generally recites the multiple event streams comprise event data from a plurality of content delivery services in the CDN, the plurality of content delivery services including a first content delivery service. On information and belief, the accused Comcast CDN functionalities do so by employing Kafka Apache or similar software to provide event streams associated with different content delivery services:[54]

> Technically speaking, event streaming is the practice of capturing data in real-time from event sources like databases, sensors, mobile devices, cloud services, and software applications in the form of streams of events; storing these event streams durably for later retrieval; manipulating, processing, and reacting to the event streams in real-time as well as retrospectively; and routing the event streams to different destination technologies as needed. Event streaming thus ensures a continuous flow and interpretation of data so that the right information is at the right place, at the right time.

[54] https://kafka.apache.org/documentation/.

- The Kafka Streams API to implement stream processing applications and microservices. It provides higher-level functions to process event streams, including transformations, stateful operations like aggregations and joins, windowing, processing based on event-time, and more. Input is read from one or more topics in order to generate output to one or more topics, effectively transforming the input streams to output streams.

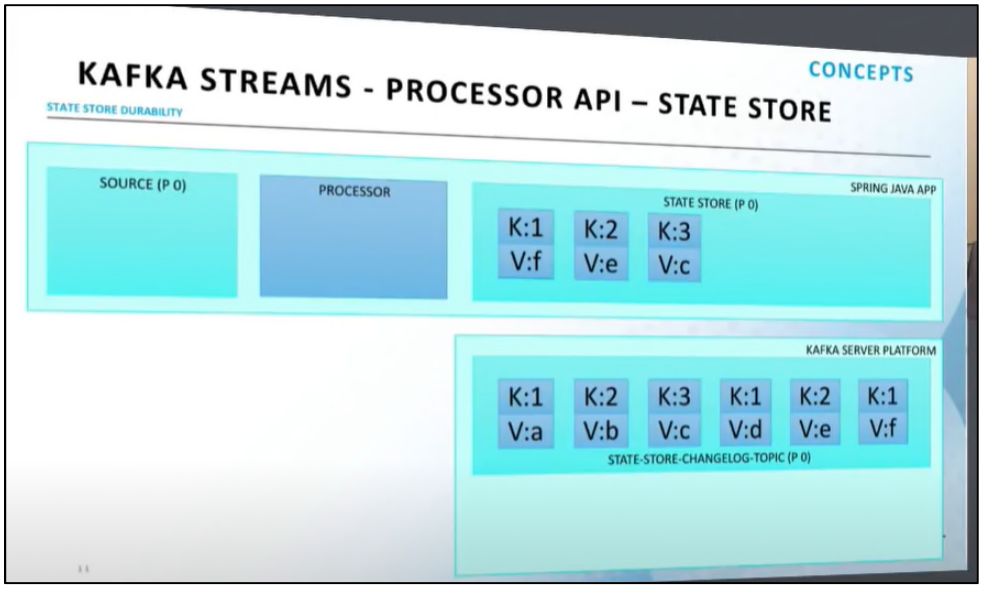
What can I use event streaming for?

Event streaming is applied to a wide variety of use cases across a plethora of industries and organizations. Its many examples include:

- To process payments and financial transactions in real-time, such as in stock exchanges, banks, and insurances.
- To track and monitor cars, trucks, fleets, and shipments in real-time, such as in logistics and the automotive industry.
- To continuously capture and analyze sensor data from IoT devices or other equipment, such as in factories and wind parks.
- To collect and immediately react to customer interactions and orders, such as in retail, the hotel and travel industry, and mobile applications.
- To monitor patients in hospital care and predict changes in condition to ensure timely treatment in emergencies.
- To connect, store, and make available data produced by different divisions of a company.
- To serve as the foundation for data platforms, event-driven architectures, and microservices.

80. The first step of the method in claim 1 further generally recites that each event of the event streams comprise a timestamp for the event and information related to the event. On information and belief, the accused Comcast CDN functionalities include or support event streams including this information. For example, the accused Comcast CDN functionalities provide event streams stored in a change log topic that provides indexes for each event with a corresponding timestamp and information relating to the event, such as a key, a value, a key-value pair, and/or optional metadata:[55]

[55] https://www.youtube.com/watch?v=oVvoqg-NiKU ("Deep Dive: Cloud Foundry Stream Processing – Monitoring Comcast's Outside Plant - Charles (Mike) Graham & Dan Carroll, Comcast Cable Communications," posted April 15, 2019 by Cloud Foundry); *see also* https://kafka.apache.org/documentation/.



Main Concepts and Terminology

An **event** records the fact that "something happened" in the world or in your business. It is also called record or message in the documentation. When you read or write data to Kafka, you do this in the form of events. Conceptually, an event has a key, value, timestamp, and optional metadata headers. Here's an example event:

- Event key: "Alice"
- Event value: "Made a payment of $200 to Bob"
- Event timestamp: "Jun. 25, 2020 at 2:06 p.m."

81. The second step of the method in claim 1 generally recites producing state data relating to and based on information represented in the event data of the multiple event streams. On information and belief, the accused Comcast CDN functionalities produce this state data by, for example, implementing Apache Kafka to produce and store timestamped state data versioned as key-value pairs:[56]

[56] https://kafka.apache.org/documentation/.

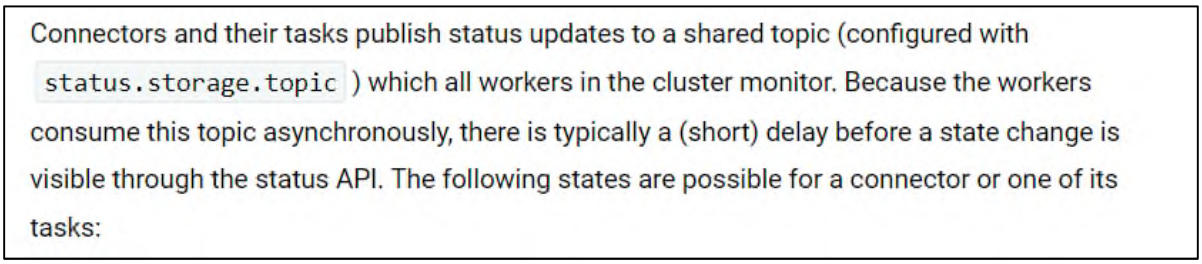
Notable changes in 3.5.0

- Kafka Streams has introduced a new state store type, versioned key-value stores, for storing multiple record versions per key, thereby enabling timestamped retrieval operations to return the latest record (per key) as of a specified timestamp. See KIP-889 and KIP-914 for more details. If the new store typed is used in the DSL, improved processing semantics are applied as described in KIP-914.

Event Sourcing

Event sourcing is a style of application design where state changes are logged as a time-ordered sequence of records. Kafka's support for very large stored log data makes it an excellent backend for an application built in this style.

82. The second step of the method in claim 1 also generally recites that, while the collector system is producing the state data, the collector system asynchronously responds to at least one query relating to the state data. On information and belief, the accused Comcast CDN functionalities asynchronously respond to at least one query by, for example, implementing Apache Kafka and responding to a query by allowing workers in a cluster to consume a topic asynchronously while simultaneously producing state data:[57]

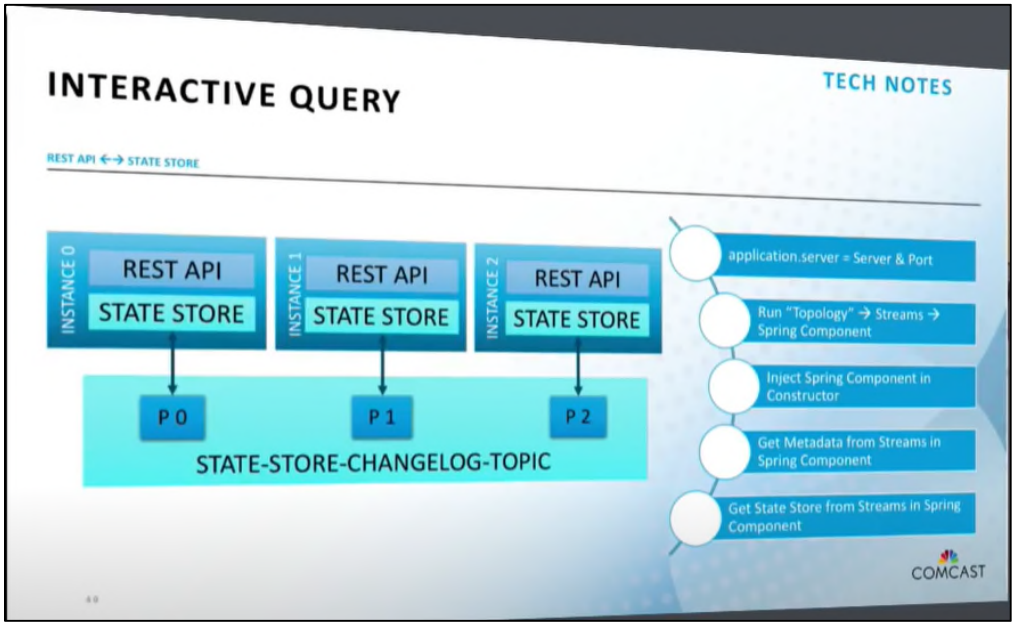


Connectors and their tasks publish status updates to a shared topic (configured with `status.storage.topic`) which all workers in the cluster monitor. Because the workers consume this topic asynchronously, there is typically a (short) delay before a state change is visible through the status API. The following states are possible for a connector or one of its tasks:

[57] https://kafka.apache.org/documentation/; *see also* https://www.youtube.com/watch?v=oVvoqg-NiKU ("Deep Dive: Cloud Foundry Stream Processing – Monitoring Comcast's Outside Plant - Charles (Mike) Graham & Dan Carroll, Comcast Cable Communications," posted April 15, 2019 by Cloud Foundry).

- **UNASSIGNED:** The connector/task has not yet been assigned to a worker.
- **RUNNING:** The connector/task is running.
- **PAUSED:** The connector/task has been administratively paused.
- **STOPPED:** The connector has been stopped. Note that this state is not applicable to tasks because the tasks for a stopped connector are shut down and won't be visible in the status API.
- **FAILED:** The connector/task has failed (usually by raising an exception, which is reported in the status output).
- **RESTARTING:** The connector/task is either actively restarting or is expected to restart soon

Not all use cases require such strong guarantees. For uses which are latency sensitive we allow the producer to specify the durability level it desires. If the producer specifies that it wants to wait on the message being committed this can take on the order of 10 ms. However the producer can also specify that it wants to perform the send completely asynchronously or that it wants to wait only until the leader (but not necessarily the followers) have the message.



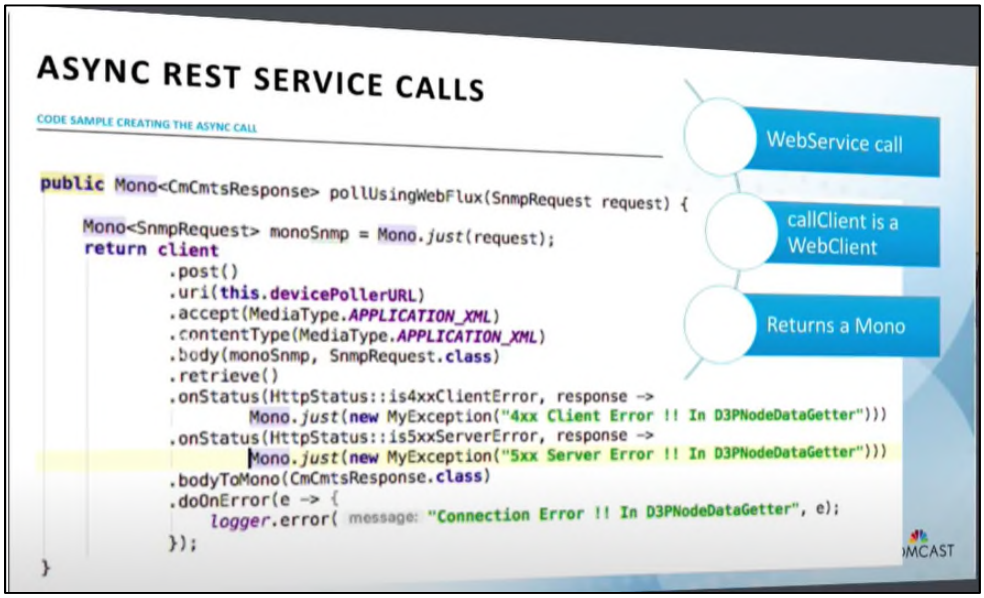
83. The third step of the method in claim 1 generally recites, in response to a query (relating to state data) from the first content delivery service, providing at least some of the state data to the first content delivery service. On information and belief, the accused Comcast CDN functionalities provide at least some of the state data to the first content delivery service by, for example, implementing Apache Kafka to provide a response to a Web Service call that relates to

state data the first CD service:[58]

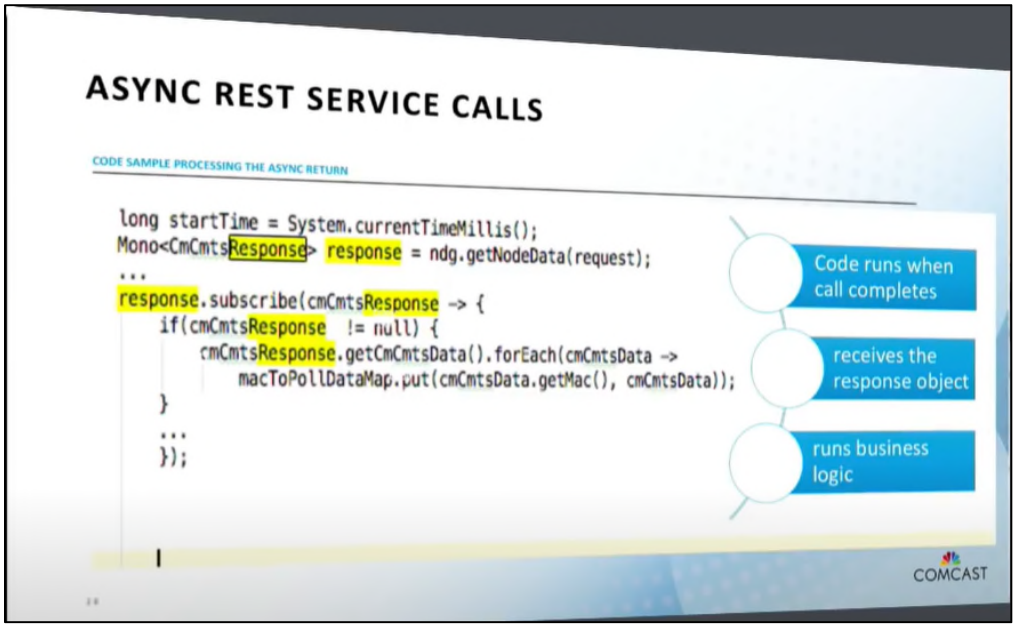
You can see the current state of OS memory usage by doing

```
$ cat /proc/meminfo
```

The meaning of these values are described in the link above.

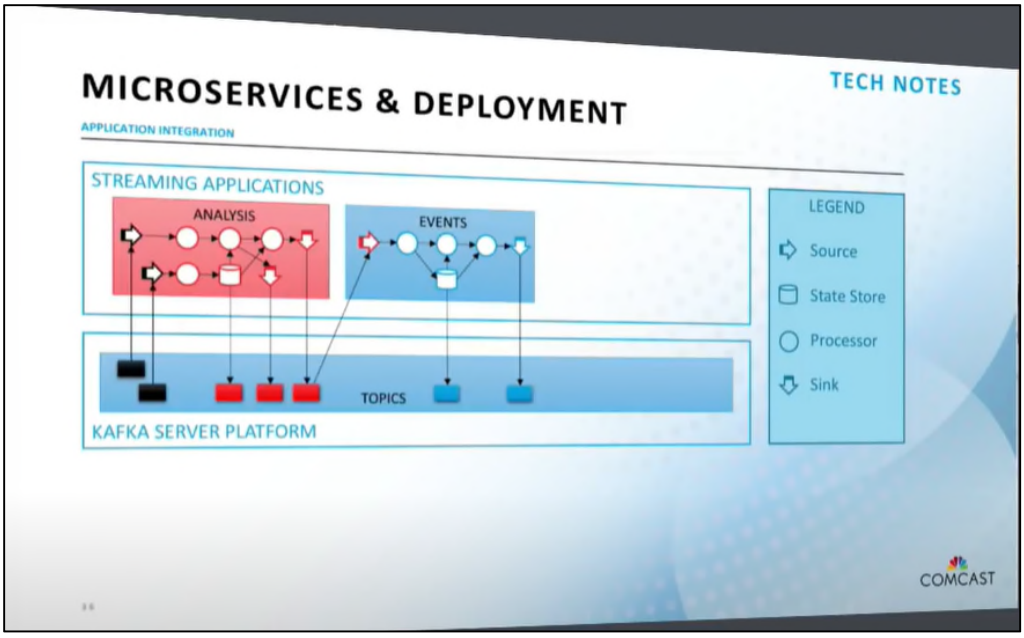


[58] https://kafka.apache.org/documentation/; *see also* https://www.youtube.com/watch?v=oVvoqg-NiKU ("Deep Dive: Cloud Foundry Stream Processing – Monitoring Comcast's Outside Plant - Charles (Mike) Graham & Dan Carroll, Comcast Cable Communications," posted April 15, 2019 by Cloud Foundry).



84. The third step of the method in claim 1 also generally recites the state data to be used to inform a peering policy of a set of peer catches. On information and belief, the accused Comcast CDN functionalities uses state data to inform a peering policy of a set of peer catches by, for example, employing Apache Kafka to define a topology whereby an application is broken up into two independently deployed applications causing two hardware components to be separated across the respective application boundaries based on state data informing a peering policy.[59] In this example, the topic shared between these two independently deployed application includes state data that is used to inform a peering policy of the peer caches:[60]

[59] https://www.youtube.com/watch?v=oVvoqg-NiKU ("Deep Dive: Cloud Foundry Stream Processing – Monitoring Comcast's Outside Plant - Charles (Mike) Graham & Dan Carroll, Comcast Cable Communications," posted April 15, 2019 by Cloud Foundry).
[60] *See id.*



Further in one implementation, the accused Comcast CDN functionalities employ Apache Kafka to create a policy (*e.g.*, Pdflush) for maintaining data on a page cache, and the policy is configurable based on state data:[61]

> In Linux, data written to the filesystem is maintained in pagecache until it must be written out to disk (due to an application-level fsync or the OS's own flush policy). The flushing of data is done by a set of background threads called pdflush (or in post 2.6.32 kernels "flusher threads").
>
> Pdflush has a configurable policy that controls how much dirty data can be maintained in cache and for how long before it must be written back to disk. This policy is described here. When Pdflush cannot keep up with the rate of data being written it will eventually cause the writing process to block incurring latency in the writes to slow down the accumulation of data.

COUNT III: INFRINGEMENT OF THE ’903 PATENT

85. Plaintiff hereby incorporates by reference each of the allegations in the foregoing paragraphs as though fully set forth herein, and further alleges as follows.

[61] https://kafka.apache.org/documentation/.

86. As explained above, in the context of CDN technology, issues arise relating to delivering content from more than one content provider. Certain CDN providers face challenges with respect to serving content associated with multiple sources. For example, issues relating to load balancing and traffic congestion at the origin servers arise for CDN providers. Embodiments of the '903 Patent address such issues and include, for example, requests for a first resource located on a first origin server being directed, based at least in part on a first alias name, to at least one repeater server.

87. Taking advantage of these technical improvements described in the '903 Patent, the accused Comcast CDN technologies, including, for example, ATC and Apache Traffic Server, have employed and continue to employ the claimed subject matter of at least claim 28 of the '903 Patent. Comcast's infringement has caused damages for which Plaintiff is entitled to compensation pursuant to 35 U.S.C. § 284

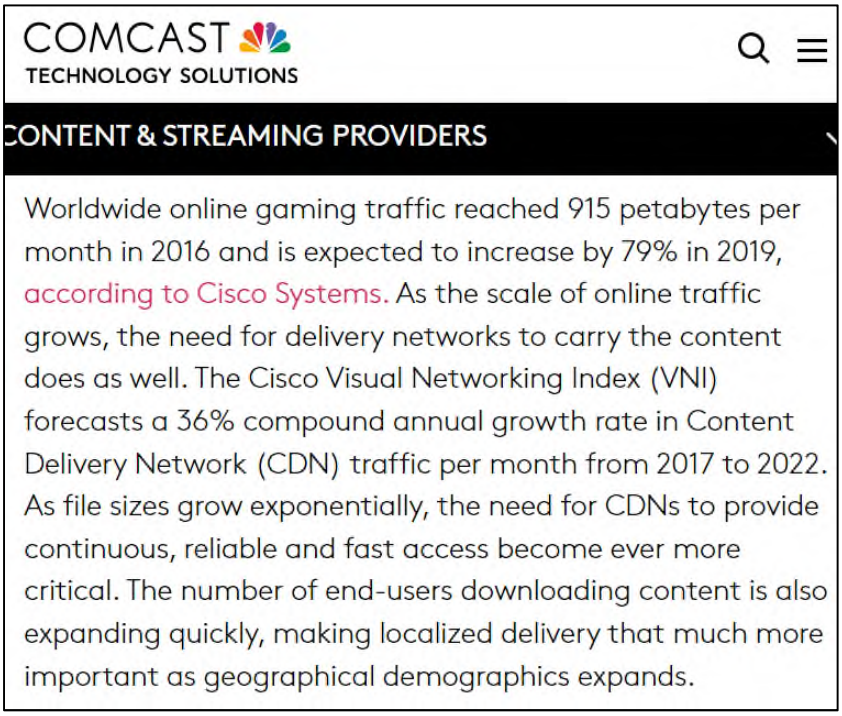
88. Claim 28 of the '903 Patent claims a method for delivery using alias names and shared repeater servers to manage requests. The accused Comcast CDN functionalities provide a method for delivering content in a network.[62] For example, Comcast provides a content delivery system, delivering content for gaming, video playback, and other purposes, including by providing CDN solutions for CDN customers. Comcast also uses its CDN to power its X1 experience.[63]

89. Although not believed to be limiting on the claim scope, the preamble of Claim 28 recites a content delivery system with a plurality of origin servers, each of said origin servers

[62] Although Sandpiper cites to claim 1, Sandpiper does not concede that any claim is representative of the entire '903 Patent nor does Sandpiper submit that Comcast is only infringing this claims. Rather, Sandpiper believes that claims in the '903 Patent are patentably distinct and the claims of the '903 Patent support alternative or additional infringement theories.

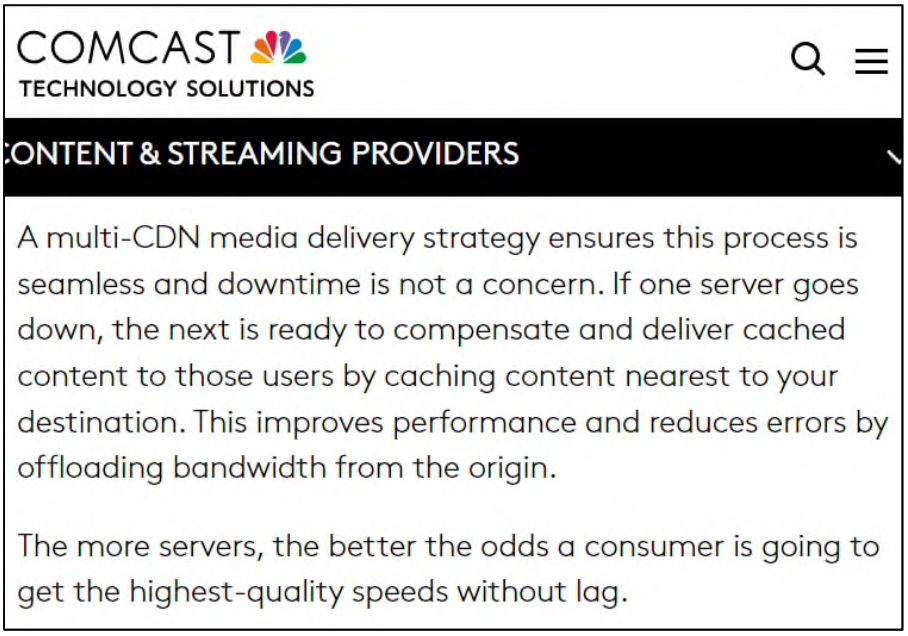
[63] https://www.comcasttechnologysolutions.com/sites/default/files/2017-10/CDN%20One%20Sheet.pdf

having resources associated therewith, and at least one shared repeater server operable to replicate resources associated with the plurality of origin servers. The accused Comcast CDN functionalities meet these limitations at least because they include origin servers with resources, and at least one repeater server is operable to replicate resources associated with the origin servers. For example, origin servers are associated with the resources (*e.g.*, video files or video game files) of the content sources for the accused Comcast CDN functionalities.[64] As one example, see "Your Origin or Ours," indicating various resources are associated with origin servers.[65]

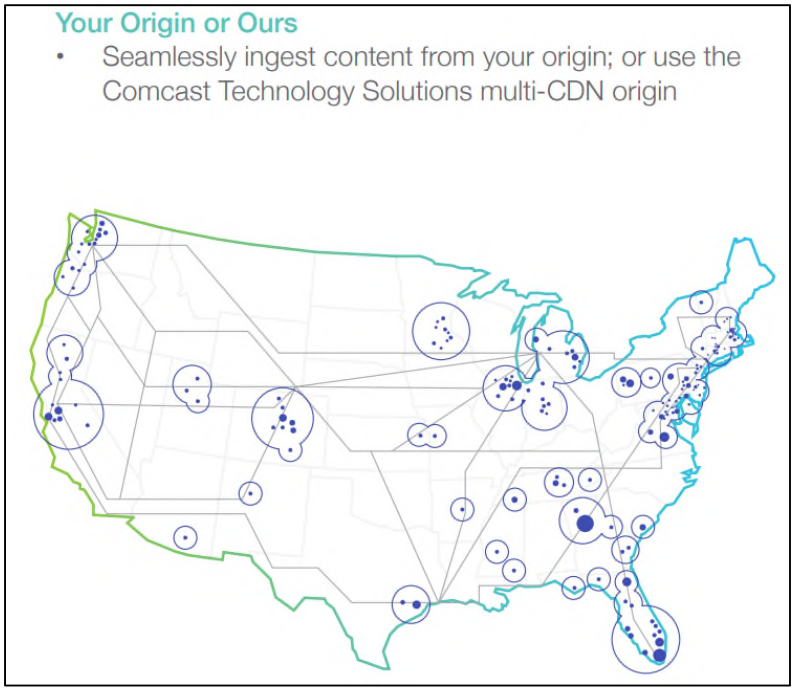


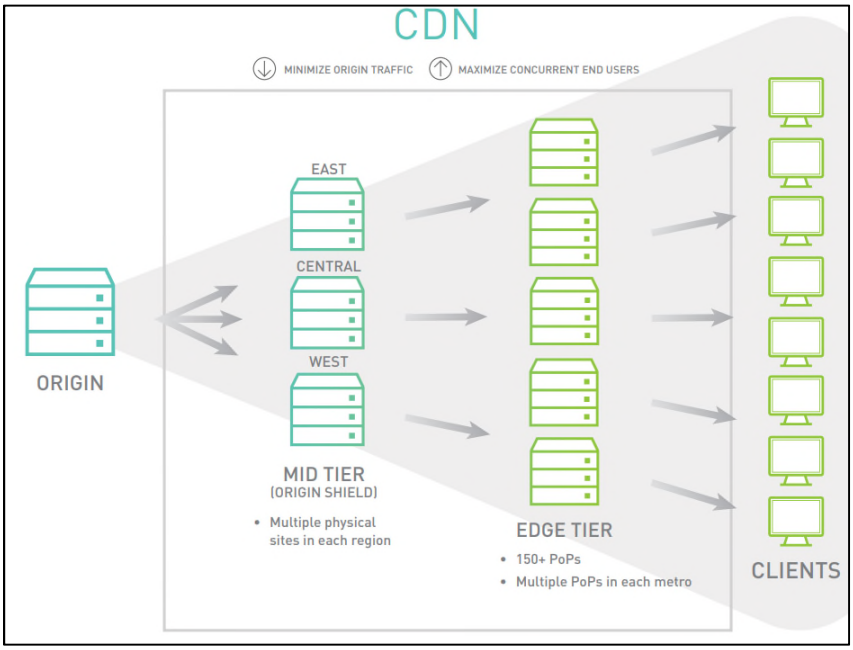
64 https://www.comcasttechnologysolutions.com/blog/cdn-media-delivery-gaming-and-virtual-reality; https://www.comcasttechnologysolutions.com/blog/why-mobile-should-drive-your-cdn-strategy.

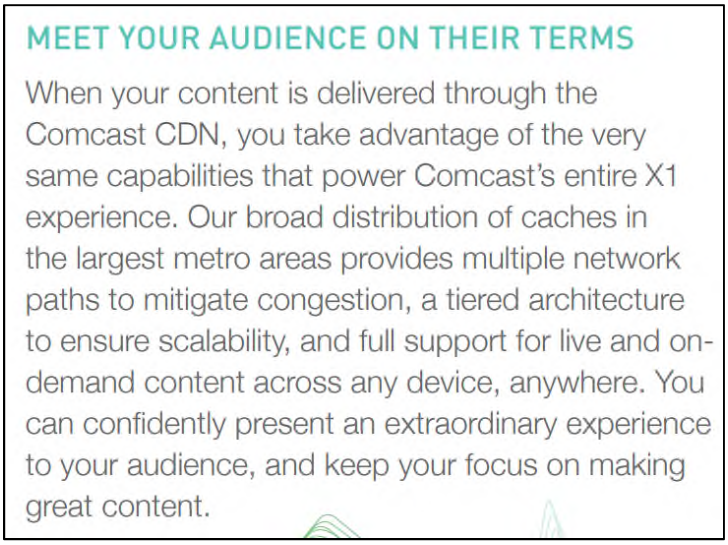
65 https://www.comcasttechnologysolutions.com/sites/default/files/2020-01/Comcast-CDN_OneSheet_01092020.pdf.


COMCAST
TECHNOLOGY SOLUTIONS
ONTENT & STREAMING PROVIDERS
A multi-CDN media delivery strategy ensures this process is seamless and downtime is not a concern. If one server goes down, the next is ready to compensate and deliver cached content to those users by caching content nearest to your destination. This improves performance and reduces errors by offloading bandwidth from the origin.
The more servers, the better the odds a consumer is going to get the highest-quality speeds without lag.


COMCAST
TECHNOLOGY SOLUTIONS
ONTENT & STREAMING PROVIDERS
Avoid the additional resources required to manage multiple partners independently, and leverage Comcast's multi-CDN fabric to deliver your broadcast quality content to any device. Take advantage of the Comcast CDN and other pure-play CDNs using DLVR multi-CDN routing technology with direct interconnects to more than 250 global networks.


Your Origin or Ours
• Seamlessly ingest content from your origin; or use the Comcast Technology Solutions multi-CDN origin


CDN
MINIMIZE ORIGIN TRAFFIC
MAXIMIZE CONCURRENT END USERS
ORIGIN
EAST
CENTRAL
WEST
MID TIER
(ORIGIN SHIELD)
• Multiple physical sites in each region
EDGE TIER
• 150+ PoPs
• Multiple PoPs in each metro
CLIENTS



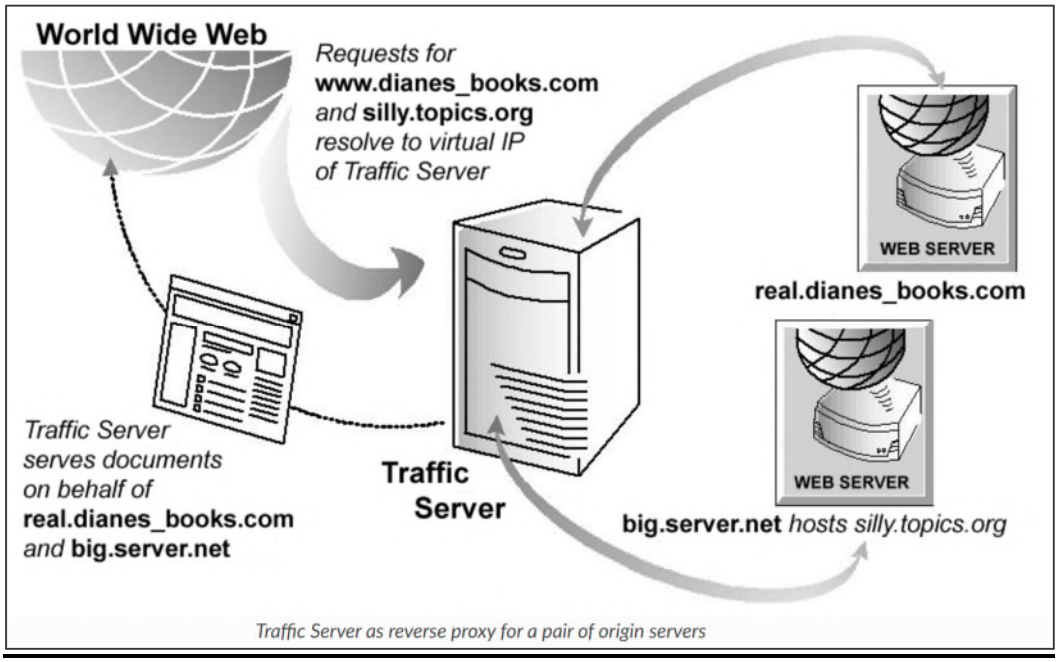
90. The first step of claim 28 generally recites associating at least one repeater server with a first alias name. The accused Comcast CDN functionalities provide at least one repeater server associated with an alias name, for example "multiple CDN cache[]" servers that handle traffic, as described below. *See id.*

91. Additionally, claim 28 recites that requests for a first resource located on a first origin server are directed, based at least in part on said first alias name, to the at least one repeater server for delivery of the first resource from said at least one repeater server. In this same example below, requests for a resource on an origin server are directed, based at least in part on an alias name, to a repeater server. For instance, the accused Comcast CDN functionalities provide repeater servers associated with an alias in order to direct requests for its CDN customers' resources to cache servers, so that "content is delivered consistently to end-users and a customer's origin load remains at an absolute minimum," as shown below.[66,67]

[66] https://www.comcasttechnologysolutions.com/node/761.
[67] https://docs.trafficserver.apache.org/en/9.2.x/admin-guide/configuration/redirecting-http-requests.en.html#reverse-proxy-and-http-redirects

3. **Strategic Absorption** – For unanticipated traffic spikes, Comcast absorbs the excess traffic by automatically spreading traffic to multiple CDN caches in the same metro. Comcast has over 100 separate caching locations in the U.S., and spreads its CDN traffic to multiple proximate physical locations in each major metro without a tradeoff in performance. These CDN locations are strategically deployed across the U.S., which (1) enables delivery in large metro areas and (2) more evenly spreads demand to prevent congestion that commonly impacts high throughput HD video and large file downloads. In addition, the Comcast network is well interconnected with many other networks. This enables high-quality content distribution, and allows content to be localized for distribution just as effectively.

4. **Reliable Service** – Many CDNs don't provide enough overhead capacity or standard origin protection tiers to absorb large traffic spikes. These practices further compound the congestion problems experienced during large events. Excess traffic is either bottlenecked in the CDN or passed back to the customer's origin, which makes end-user problems worse. This does not happen with Comcast's CDN, which has a QoS variance that is 20-30% better than other CDNs based on objective third-party data. Whether it's serving high seasonal traffic peaks or one-time massive scale video events, content is delivered consistently to end-users and a customer's origin load remains at an absolute minimum.



Traffic Server as reverse proxy for a pair of origin servers

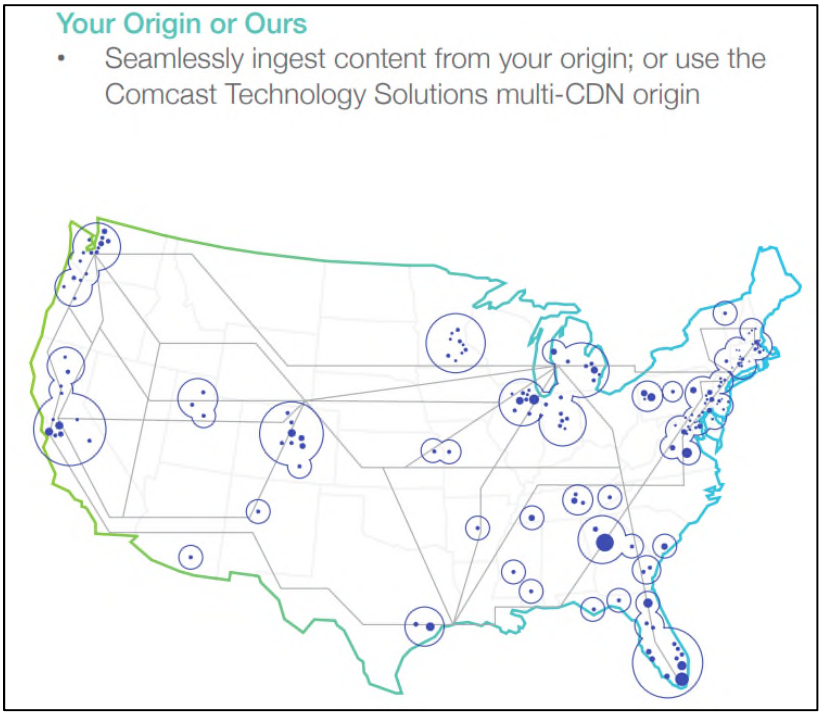
Traffic Server can accept requests on behalf of the origin server and improve the speed and quality of web serving by reducing load and hot spots on backup origin servers. For example, a web host can maintain a scalable Traffic Server system with a set of low-cost, low-performance, less-reliable PC origin servers as backup servers. In fact, a single Traffic Server can act as the virtual origin server for multiple backup origin servers, as shown in the figure below.

92. The second step of claim 28 generally recites associating at least one repeater server with a second alias name. The accused Comcast CDN functionalities associate the repeater server with a second alias name, for example an alias name associated with another Comcast CDN content source.

93. Additionally, claim 28 generally recites that that requests for a second resource located on a second origin server are directed, based at least in part on said second alias name, to the at least one repeater server for delivery of the second resource from said at least one repeater server, wherein the second origin server is distinct from the first origin server. Comcast provides that requests for a second resource are directed, based at least in part on the second alias name, to the repeater server. As one example, the accused Comcast CDN functionalities use an alias associated with first CDN content source, and an alias associated with a second CDN content source, to direct requests to a repeater server, for example, using one of the "multiple CDN cache[]" servers that handle traffic. As another example, the accused Comcast CDN functionalities use an alias associated with a first origin and an alias associated with a second origin (*e.g.*, a different resource for the same content source) to direct requests to a repeater server. In this example, a Comcast CDN content source may have resources associated with more than one origin.[68] The accused Comcast CDN functionalities include "multiple CND caches" comprising repeater server(s), for example.[69]

[68] https://www.comcasttechnologysolutions.com/sites/default/files/2020-01/Comcast-CDN_OneSheet_01092020.pdf.

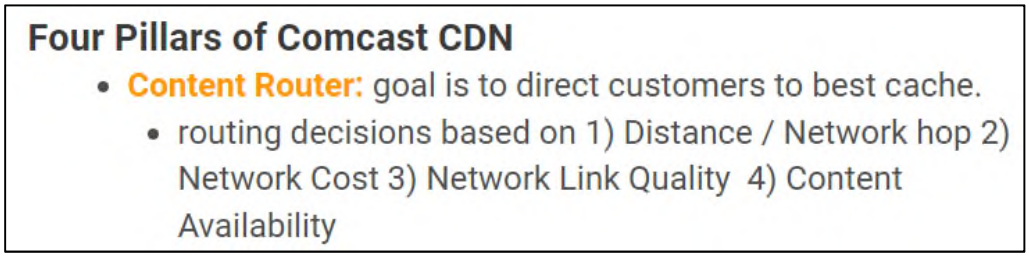
[69] https://www.comcasttechnologysolutions.com/node/761.

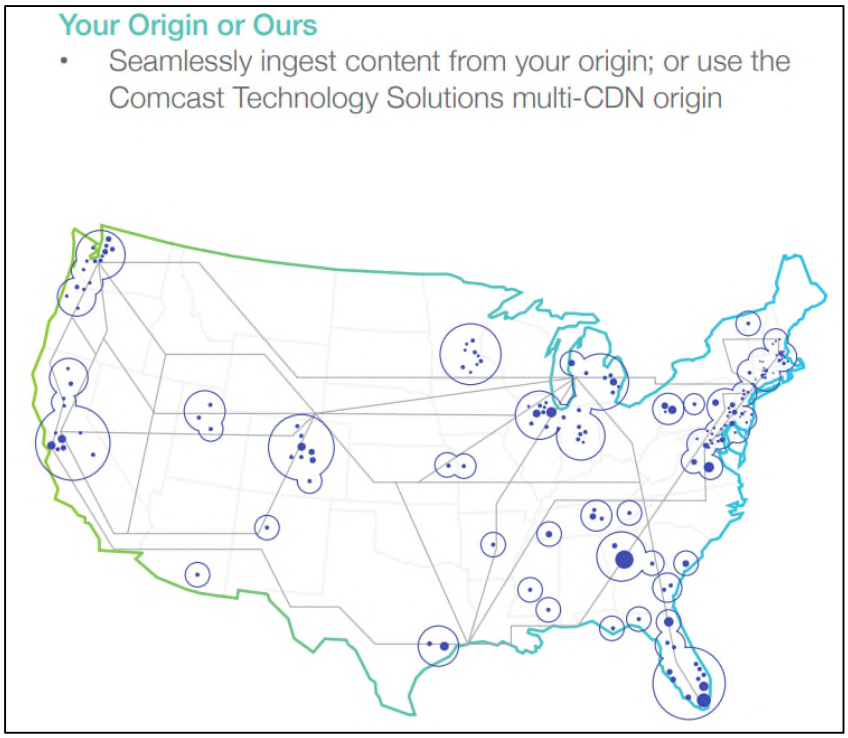


3. **Strategic Absorption** – For unanticipated traffic spikes, Comcast absorbs the excess traffic by automatically spreading traffic to multiple CDN caches in the same metro. Comcast has over 100 separate caching locations in the U.S., and spreads its CDN traffic to multiple proximate physical locations in each major metro without a tradeoff in performance. These CDN locations are strategically deployed across the U.S., which (1) enables delivery in large metro areas and (2) more evenly spreads demand to prevent congestion that commonly impacts high throughput HD video and large file downloads. In addition, the Comcast network is well interconnected with many other networks. This enables high-quality content distribution, and allows content to be localized for distribution just as effectively.

94. The third step of claim 28 generally recites a table listing origin servers having content located thereon, wherein said content is authorized for delivery to client machines via the at least one shared repeater server. The accused Comcast CDN functionalities provide a table listing origin servers. As one example, the accused Comcast CDN functionalities provide information relating origin servers to cached content, in order to route requests and/or to serve

cached content from origin servers.[70] On information and belief, said content is authorized for delivery to client machines via the at least one shared repeater server.





95. The final step of claim 28 of the ’903 patent generally recites at least one repeater server further constructed and adapted to analyze, using the table, an alias name received with a client request for a particular resource to determine an origin server associated with the particular resource. The accused Comcast CDN functionalities meets this limitation. For example, the accused Comcast CDN functionalities include multiple CDN cache server(s) that analyze an alias

[70] https://www.bizety.com/2015/07/15/deep-dive-comcast-cdn-architecture/; https://www.comcasttechnologysolutions.com/sites/default/files/2020-01/Comcast-CDN_OneSheet_01092020.pdf.

name to determine an origin server associated with a resource. In one example, a resource (such as a property) of a CDN content source is associated with an origin server, and a repeater server is constructed to analyze an alias to determine the origin server.

COUNT IV: INFRINGEMENT OF THE '322 PATENT

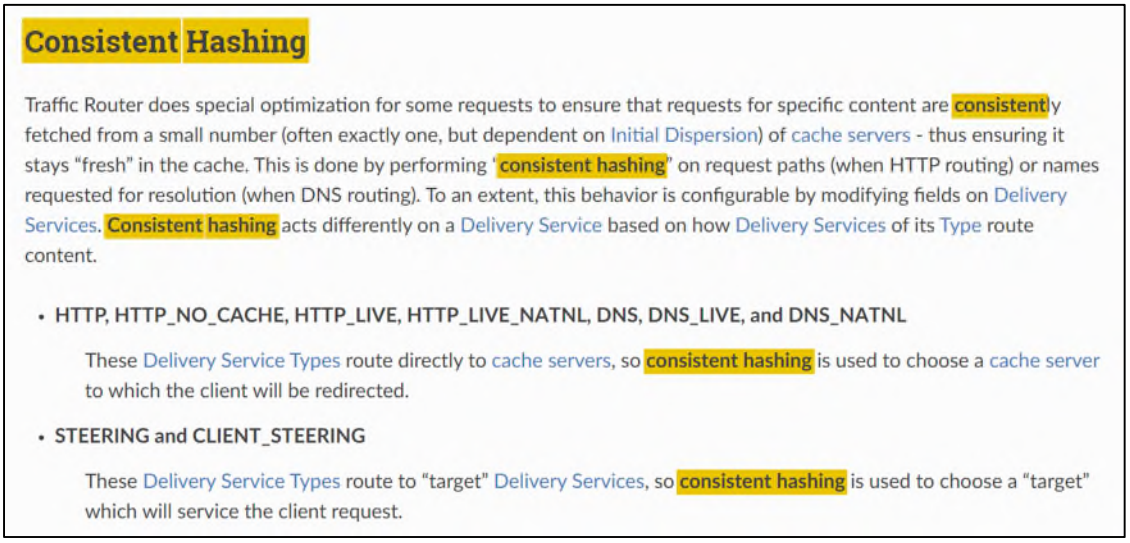
96. Plaintiff hereby incorporates by reference each of the allegations in the foregoing paragraphs as though fully set forth herein, and further alleges as follows.

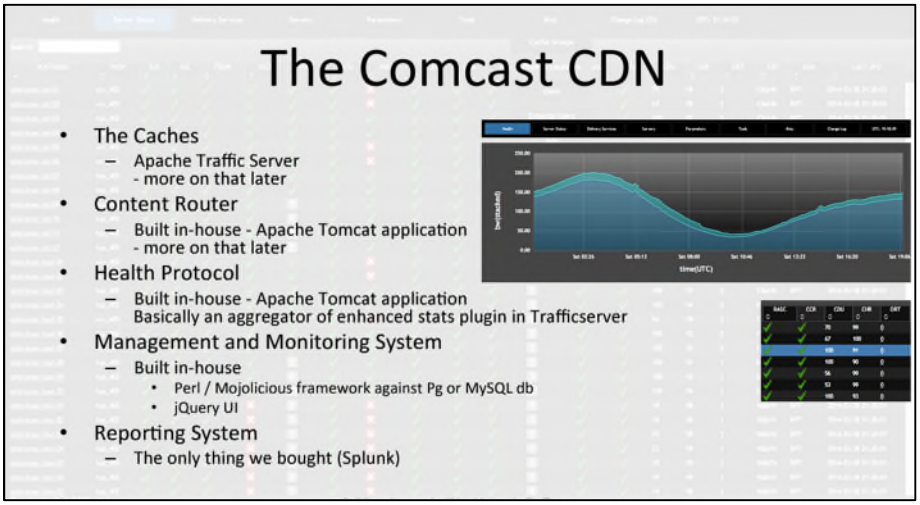
97. Taking advantage of the technical improvements described in the '322 Patent, the accused Comcast CDN technologies have, pursuant to 35 U.S.C. § 271, infringed and continue to infringe the claimed method for rewriting requests and/or responses claimed in at least claim 11 of the '322 Patent.[71] Comcast's infringement has caused and will continue to cause damages for which Plaintiff is entitled to compensation pursuant to 35 U.S.C. § 284.

98. Claim 11 of the '322 Patent claims a method for handling data requests in a distributed data delivery network. On information and belief, the accused Comcast CDN functionalities employ caching software, including, for example, ATC and Apache Traffic Server, to provide a distributed data delivery network that handles data requests:[72]

[71] Although Sandpiper cites to claims 11 and 16, Sandpiper does not concede that any claim is representative of the entire '322 Patent nor does Sandpiper submit that Comcast is only infringing these claims. Rather, Sandpiper believes that claims in the '322 Patent are patentably distinct and the claims of the '322 Patent support alternative or additional infringement theories.

[72] https://github.com/apache/trafficcontrol; https://traffic-control-cdn.readthedocs.io/en/latest/admin/traffic_router.html#consistent-hashing; http://events17.linuxfoundation.org/sites/events/files/slides/apachecon_jvd_2014_v2_16x9.pdf.





99. The first step of the method in claim 11 of the '322 Patent generally recites intercepting and modifying a data resource request from a user requesting data from the network

prior to the data resource request being advanced by a first data server in the network to create a modified data resource request. On information and belief, the accused Comcast CDN functionalities perform this step, for example, where ATC and Apache Traffic Server intercepts a client HTTP request, rewrites the HTTP header to modify the client request, and then advances the request to another cache server or origin server for further processing.

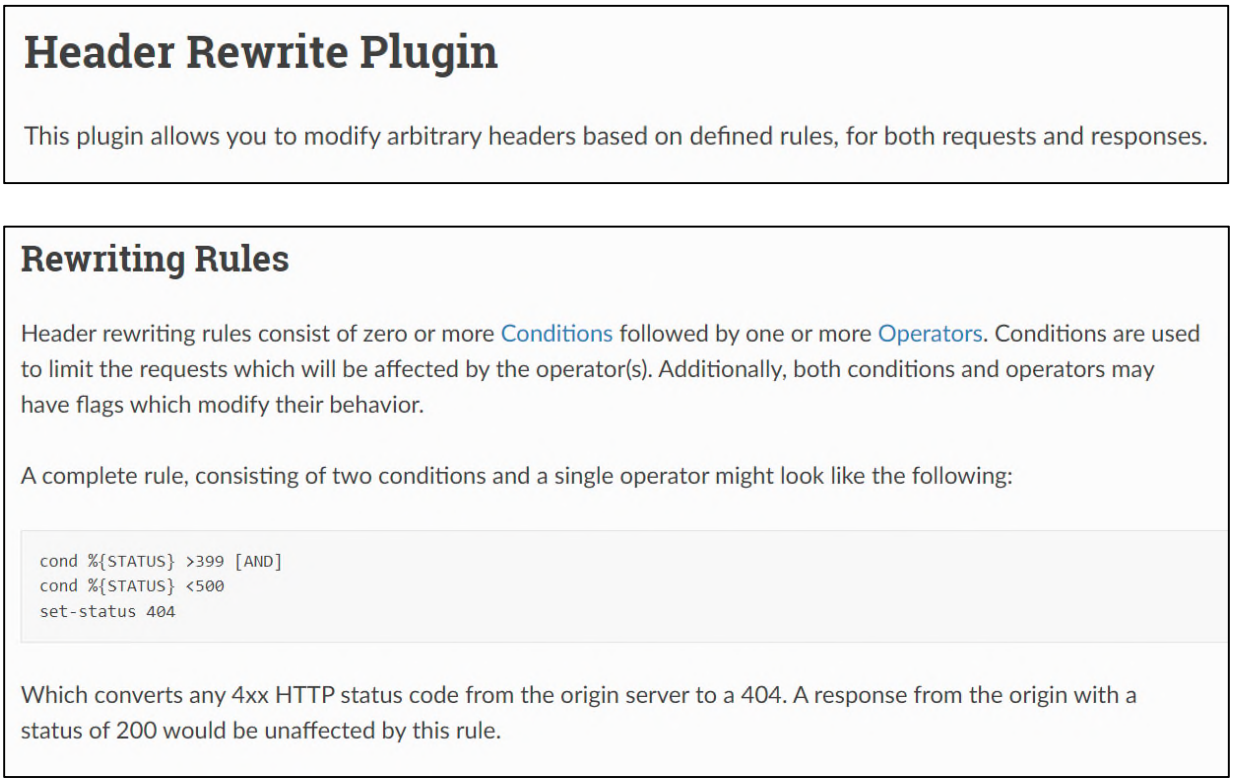
100. For example, the ATC source code details how ATC calculates a consistent hash from the HTTP request and then inserts a preferred route Uniform Resource Locator (URL) (based on the calculated hash) into the HTTP response header that is sent to the client for multi route or single route requests.[73] As another example, the Apache Traffic Server source code provides that the Apache Traffic Server is initialized, receives the HTTP request, uses the HTTP header rewriter plugin to rewrite the HTTP request header, and then sends the rewritten HTTP request header to the parent tier cache server or origin server.[74]

---

[73] *See* trafficcontrol\traffic_router\core\src\main\java\org\apache\traffic_control\traffic_router\core\http\RouterFilter.java (detailing how the ATC code receives an HTTPRequest from a user and invokes, among other functions, the *writeHttpResponse( )* function to set the route result to the HTTP header).

[74] *See* trafficserver\src\traffic_server\traffic_server.c and trafficserver\src\proxy\http\HttpProxyServerMain.cc (detailing how the Apache Traffic Service invokes the *main()* function, and subsequent functions (*e.g.*, *start_HttpProxyServer( )*) to configure an HTTP proxy server to become open to traffic and begin event processing); *see also* trafficserver\src\proxy\http\HttpTransact.cc (detailing the *HttpTransact::handleRequest( )* function and the *TRANSACT_RETURN* function, which sets up the server to intercept HTTP requests by creating an API hook that the rewrite header plugin can access) and trafficserver\plugins\header_rewrite\header_rewrite.cc (detailing the server calling the *TSRemapDoRemap( )* for each request sent to the header rewrite plugin, which determines a rule to apply to the hook via an *exec( )* function (*e.g.*, trafficserver\plugins\header_rewrite\operators.cc)); *see also* trafficserver\src\proxy\http\HttpTransact.cc and trafficserver\src\proxy\http\HttpSM.cc (detailing how Apache Traffic Server utilizes the *handleRequest( )* and *do_cache_lookup_and_read( )* functions to advance a HTTP request with modified header information to a parent (mid-tier) cache or to the origin server when the chache is not found in the edge cache server).

101. Further, on information and belief, the following information provides additional examples for how ATC intercepts HTTP requests from user, controls the routing of the HTTP request to a cache server based on a calculated consistent hash of the request using data provided in the request, and then writes the consistent hash into the header of the HTTP response which is sent to the user for redirecting the request to the identified Cache Server (*e.g.*, the Apache Traffic Server):[75]



Header Rewrite Plugin

This plugin allows you to modify arbitrary headers based on defined rules, for both requests and responses.

Rewriting Rules

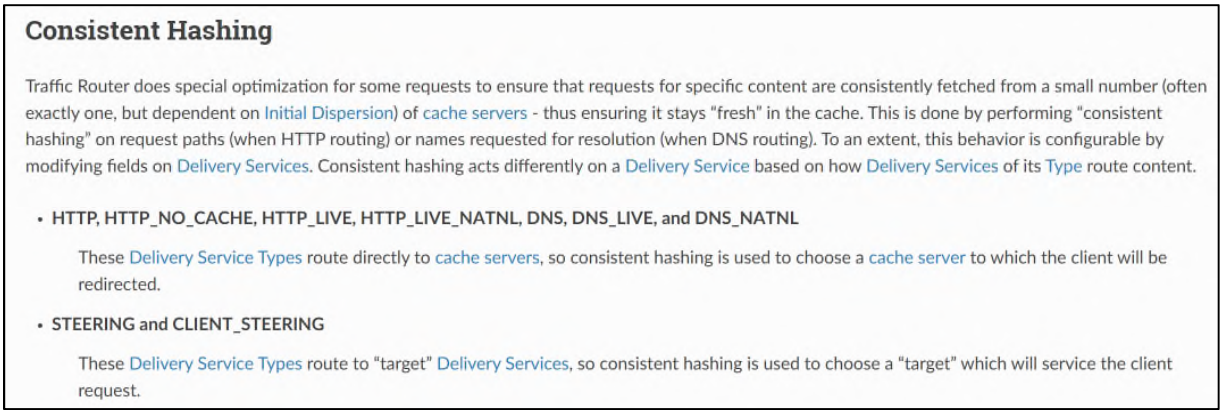
Header rewriting rules consist of zero or more Conditions followed by one or more Operators. Conditions are used to limit the requests which will be affected by the operator(s). Additionally, both conditions and operators may have flags which modify their behavior.

A complete rule, consisting of two conditions and a single operator might look like the following:

```
cond %{STATUS} >399 [AND]
cond %{STATUS} <500
set-status 404
```

Which converts any 4xx HTTP status code from the origin server to a 404. A response from the origin with a status of 200 would be unaffected by this rule.

[75] https://docs.trafficserver.apache.org/en/latest/admin-guide/plugins/header_rewrite.en.html.



Consistent Hashing

Traffic Router does special optimization for some requests to ensure that requests for specific content are consistently fetched from a small number (often exactly one, but dependent on Initial Dispersion) of cache servers - thus ensuring it stays "fresh" in the cache. This is done by performing "consistent hashing" on request paths (when HTTP routing) or names requested for resolution (when DNS routing). To an extent, this behavior is configurable by modifying fields on Delivery Services. Consistent hashing acts differently on a Delivery Service based on how Delivery Services of its Type route content.

- **HTTP, HTTP_NO_CACHE, HTTP_LIVE, HTTP_LIVE_NATNL, DNS, DNS_LIVE, and DNS_NATNL**

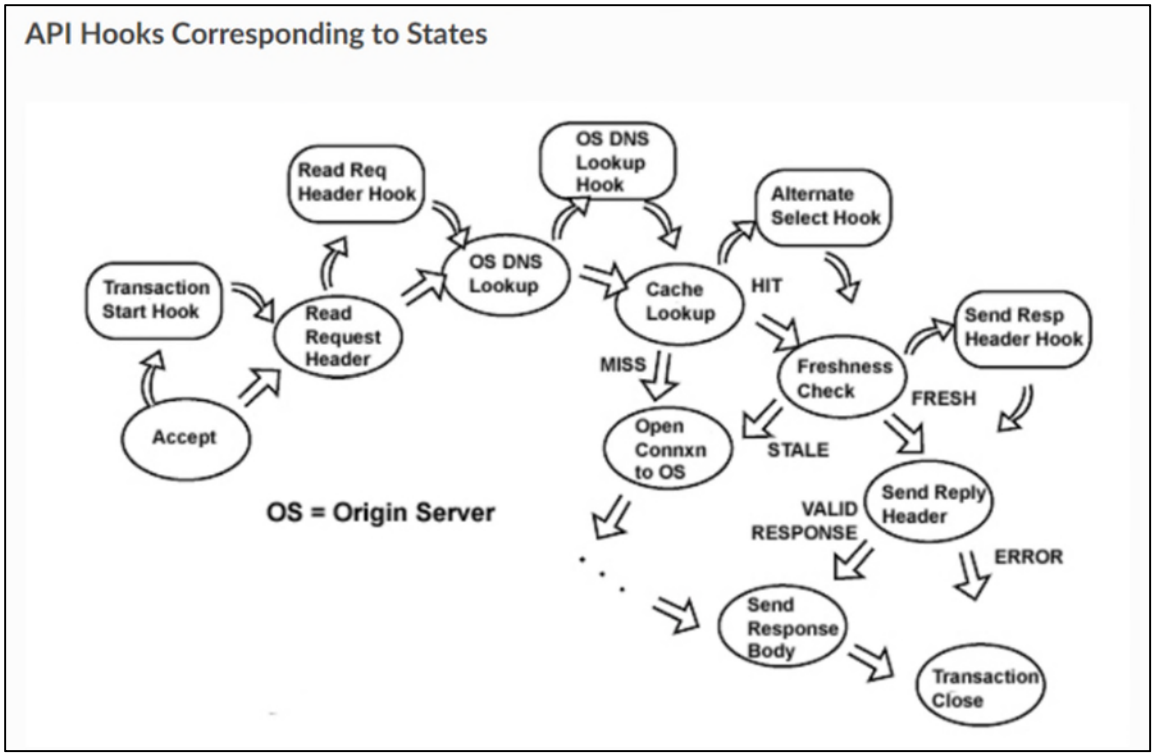
  These Delivery Service Types route directly to cache servers, so consistent hashing is used to choose a cache server to which the client will be redirected.

- **STEERING and CLIENT_STEERING**

  These Delivery Service Types route to "target" Delivery Services, so consistent hashing is used to choose a "target" which will service the client request.

102. The second step of the method in claim 11 of the '322 Patent generally recites controlling the routing of requested data from a data server in the network to the user based on the modified data resource request where the requested data remains unmodified through the delivery. On information and belief, for example, the accused Comcast CDN functionalities utilize Apache Traffic Server to build a response containing the requested data to be sent to the user if the requested data is available either in the cache server or in the origin server.[76]

[76] https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/introduction.en.html.



103. Further, on information and belief, the Apache Traffic Server source code details how it may route a request modified by the Header Rewrite plugin to a mid-tier cache server in the CDN or to the origin server (with the modified header information in the HTTP request), and, if the requested data is available either in the cache server or in the origin server, Apache Traffic Server builds a response containing the requested data to be sent to the user. On information and belief, the content (requested data) remains unmodified throughout the delivery process.[77]

COUNT V: INFRINGEMENT OF THE '692 PATENT

104. Plaintiff hereby incorporates by reference each of the allegations in the foregoing paragraphs as though fully set forth herein, and further alleges as follows.

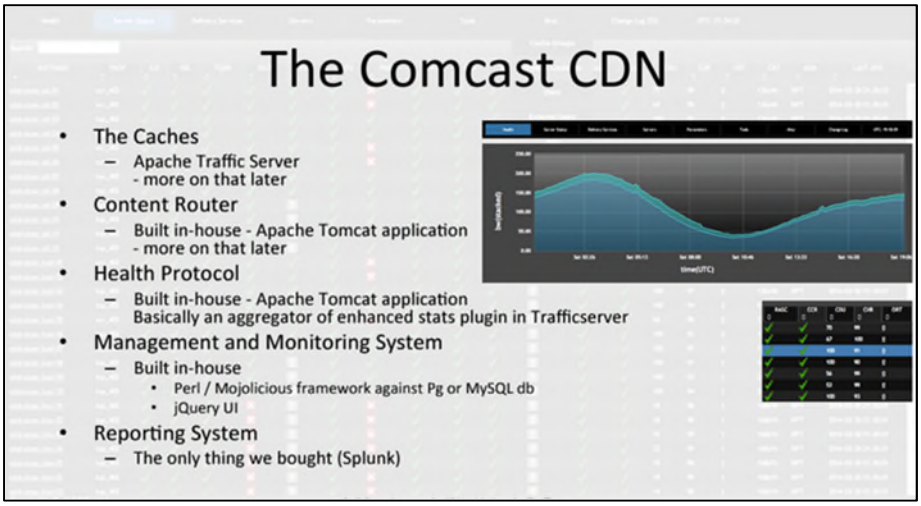
[77] *See* trafficserver\src\proxy\http\HttpTransact.cc and trafficserver\src\proxy\http\HttpSM.cc (detailing the *handleRequest( )* function, *do_cache_lookup_and_read( )* function, *HttpCacheSM::state_cache_open_read( )* function, *HandleCacheOpenRead( )* function, and *HandleCacheOpenReadHit( )* function).

105. Taking advantage of the technical improvements described in the '692 Patent, the accused Comcast CDN technologies have, pursuant to 35 U.S.C. § 271, infringed and continue to infringe the claimed method for delivering content in a CDN as claimed in at least claims 1, 2, 7, and 8 of the '692 Patent.[78] Comcast's infringement has caused and will continue to cause damages for which Plaintiff is entitled to compensation pursuant to 35 U.S.C. § 284.

106. Claim 1 of the '692 Patent claims a method for delivering content in a CDN comprising at least a first tier of servers. On information and belief, the accused Comcast CDN functionalities deliver content in a CDN that includes tiers of servers, for example edge-tier or mid-tier cache servers. As one example, Comcast CDN employs Apache Traffic Server as a caching server for its CDN. In this example, Apache Traffic Server comprises at least one origin server and multiple mid-tier and edge-tier cache servers, which replicate content from the origin server and provide the content to client machines:[79]

[78] Although Sandpiper cites to claim 1, 2, 7, and 8 of the '692 Patent Sandpiper does not concede that any claim is representative of the entire '692 Patent nor does Sandpiper submit that Comcast is only infringing these claims. Rather, Sandpiper believes that claims in the '692 Patent are patentably distinct and the claims of the '692 Patent support alternative or additional infringement theories.

[79] http://events17.linuxfoundation.org/sites/events/files/slides/apachecon_jvd_2014_v2_16x9.pdf; https://traffic-control-cdn.readthedocs.io/en/v7.0.1/overview/cache_groups.html#parent; https://traffic-control-cdn.readthedocs.io/en/latest/overview/introduction.html.



Consider the example CDN in Fig. 1. Here some country/province/region has been divided into quarters: Northeast, Southeast, Northwest, and Southwest. The arrows in the diagram indicate the flow of requests. If a client in the Northwest, for example, were to make a request to the Delivery Service, it would first be directed to some cache server in the "Northwest" Edge-tier *Cache Group*. Should the requested content not be in cache, the Edge-tier server will select a parent from the "West" *Cache Group* and pass the request up, caching the result for future use. All Mid-tier *Cache Groups* (usually) answer to a single Origin that provides canonical content. If requested content is not in the Mid-tier cache, then the request will be passed up to the Origin and the result cached.

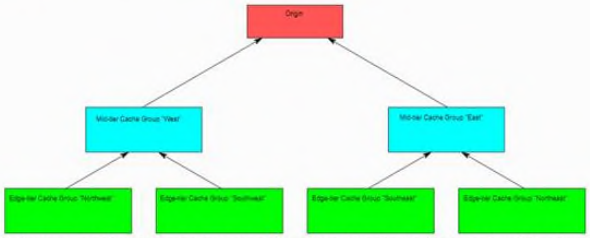
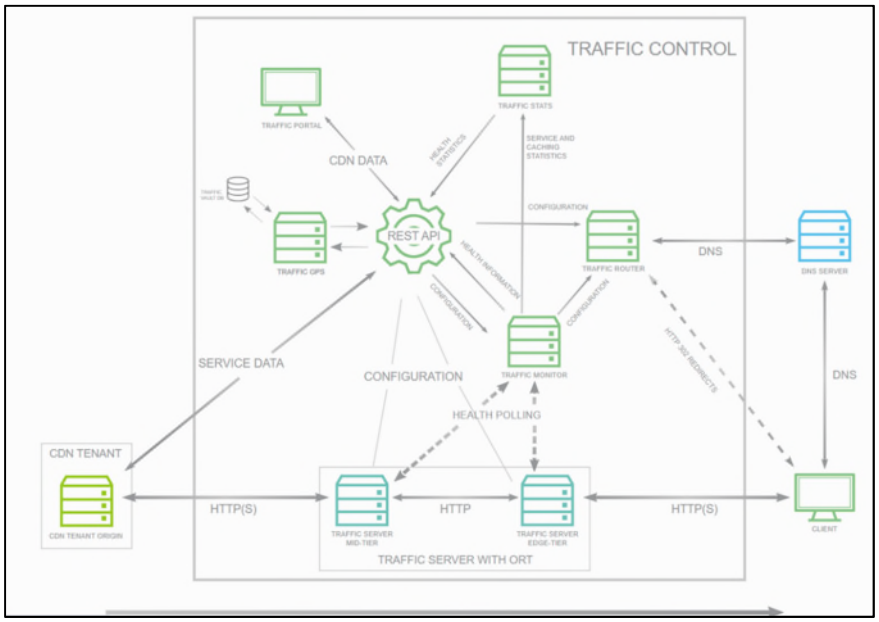
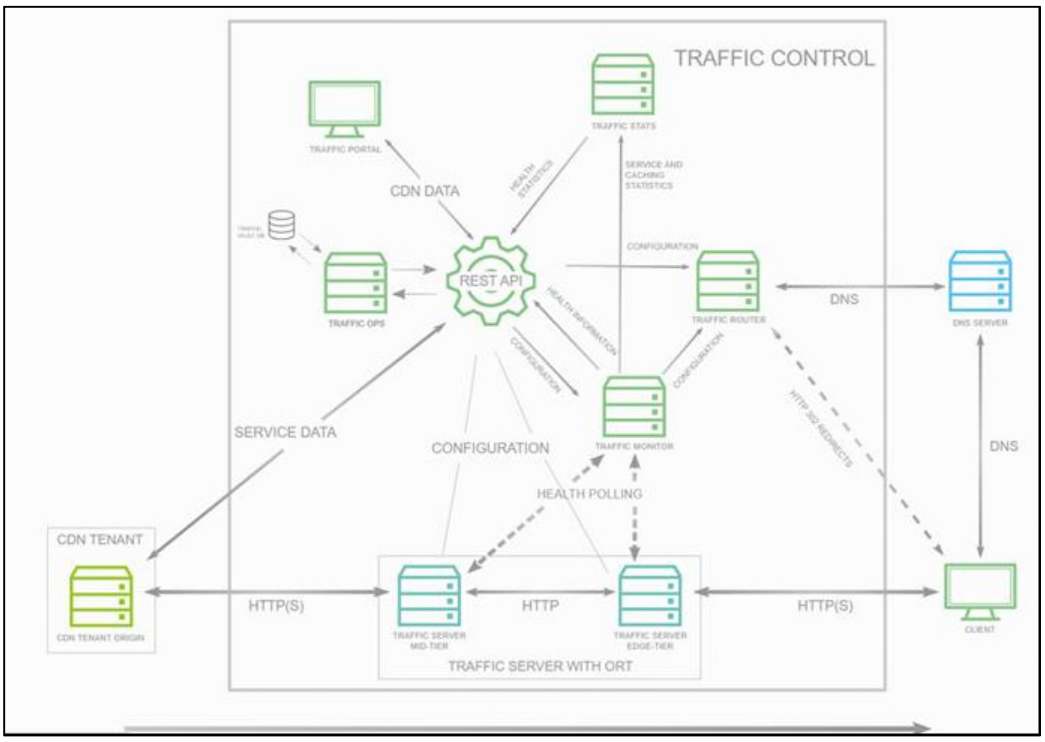


Fig. 1 *An example CDN that shows the hierarchy between four Edge-tier Cache Groups, two Mid-tier Cache Groups, and one Origin*



107. The first step of the method in claim 1 generally recites a first server in the first tier of servers to obtain a request from a client for a resource, the resource being available as part of a content provider's library. On information and belief, the accused Comcast CDN functionalities perform this step, for example, by utilizing an edge server (first server) in the edge-tier (the first tier of servers) to obtain a request from a client for a resource, where that resource is a part of the origin server resources (*e.g.*, as part of a content provider's library). In one example, Apache Traffic Server source code provides that Apache Traffic Server receives a request for content on the network at an edge server, which subsequently handles the request and processes it to locate the particular content requested by the user. In this example, if the content is unavailable on the server, the Apache Traffic Server code and technical documents provide that the network retrieves the content from the origin server or from a parent proxy server (*e.g.*, a mid-tier cache server):[80]



[80] *See* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *HandleRequest( )* and *HandleCacheOpenReadMiss( )* functions); https://docs.trafficserver.apache.org/en/latest/admin-guide/introduction.en.html.

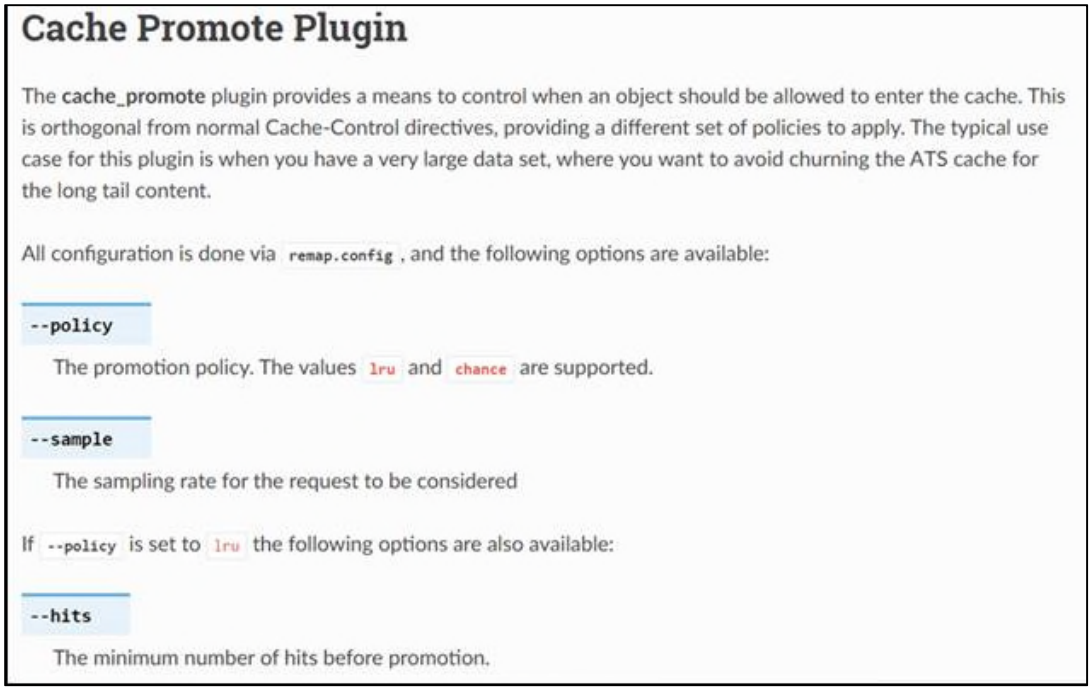
Traffic Server as a Web Proxy Cache

As a web proxy cache, Traffic Server receives user requests for web content as those requests travel to the destined web server (origin server). If Traffic Server contains the requested content, then it serves the content directly. If the requested content is not available from cache, then Traffic Server acts as a proxy: it obtains the content from the origin server on the user's behalf and also keeps a copy to satisfy future requests.

108. The second step of the method in claim 1 generally recites, if the resource is not available at the first server or at a peer of the first server, determining if the resource is popular. On information and belief, as one example, the accused Comcast CDN functionalities use the Cache Promote Plugin of Apache Traffic Server to determine if a resource is popular, if the resource is not available at an edge server (i.e. a first server), or if the resource is at a peer of an edge server. In this example, Apache Traffic Server source code details how it handles the receipt of a client request, attempts to find the requested content in a cache, and flags the instance(s) where a requested resource is not found, after which the popularity of the requested resource is determined.[81] Further to this example, Apache Traffic Server source code and technical documents detail how Apache Traffic Server uses a defined policies and algorithms to determine resource popularity.[82]

[81] *See* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *HandleRequest( )* and *HandleCacheOpenRead( )* functions).

[82] *See* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *HandleRequest(*) and *DecideCacheLookup( )* functions); *see also* trafficserver/plugins/cache_promote/cache_promote.cc (detailing the *Cont_handle_policy( )* function); *see also* trafficserver/plugins/cache_promote/lru_policy.cc (detailing the *doPromote( )* function); *see also* https://docs.trafficserver.apache.org/en/7.1.x/admin-guide/plugins/cache_promote.en.html.



Cache Promote Plugin

The **cache_promote** plugin provides a means to control when an object should be allowed to enter the cache. This is orthogonal from normal Cache-Control directives, providing a different set of policies to apply. The typical use case for this plugin is when you have a very large data set, where you want to avoid churning the ATS cache for the long tail content.

All configuration is done via `remap.config`, and the following options are available:

`--policy`

The promotion policy. The values `lru` and `chance` are supported.

`--sample`

The sampling rate for the request to be considered

If `--policy` is set to `lru` the following options are also available:

`--hits`

The minimum number of hits before promotion.

109. The third step of the method in claim 1 generally recites, if the resource is determined to be popular, then the first server obtaining the resource and the first server serving the resource to the client. On information and belief, as one example, the accused Comcast CDN functionalities use the Cache Promote Plugin to obtain the requested content and serve the content to the client if the content is determined to be popular. In this example, Apache Traffic Server source code describes how the edge server finds a server from which to fetch the requested content, update the cache with the requested content, and build a response containing to content which is sent to the client.[83]

110. The fourth step of the method in claim 1 generally recites, if the resource is determined not to be popular, directing the client to a second server in a second tier of servers distinct from the first tier of servers. On information and belief, as one example, the accused

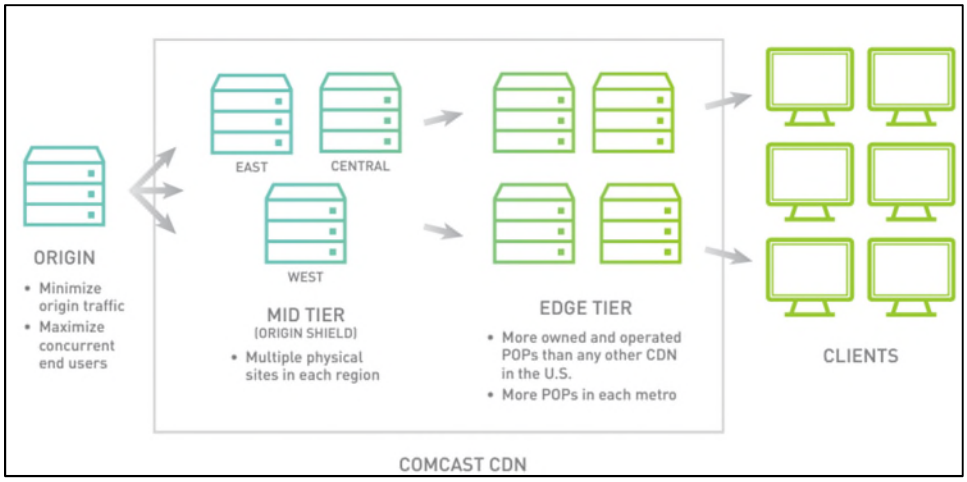
[83] *See* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *HandleCacheOpenRead( ), HandleCacheOpenReadMiss( ), HandleResponse( ),* and *handle_cache_operation_on_forward_server_response( )* functions).

Comcast CDN functionalities use the Cache Promote Plugin to determine if a resource is not popular and then direct the client to a second server in a second tier of servers that are distinct from the first tier of servers. In this example, Apache Traffic Server source code details, after determining requested content is not popular, how it will signal no cache operations will be performed for that content and create a tunnel between the responding server and the client so the other server with the content can serve the request.[84]

111. The fourth step of the method in claim 1 also generally recites, if the resource is determined not to be popular, the second server to comprise a first portion of the content provider's library, the first portion comprising at least the resource. On information and belief, as one example, the accused Comcast CDN functionalities have a mid-tier cache server (*e.g.*, a second server) that stores a portion of the origin server resources (*e.g.*, a first portion of the content provider's library). In this example, Apache Traffic Server source code and technical documents show how, when requested content is not available on a cache server, a cache miss occurs and the request is forwarded to an origin server to serve the requested content. In another example, a parent proxy sever that has the content may receive a request from a child server for the content, resulting in a cache hit occurring and delivery of the requested content.[85]

[84] *See* trafficserver/plugins/cache_promote/cache_promote.cc (detaling the *Cont_handle_policy( )* function); *see also* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *HandleCacheOpenReadMiss( ), HandleResponse( ), handle_forward_server_connection_open( ),* and *handle_cache_operation_on_forward_server_response( )* functions).

[85] *See* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *HandleCacheOpenRead( ), HandleCacheOpenReadHit( ) functions)*; *see also* https://www.comcasttechnologysolutions.com/sites/default/files/2017-10/CDN%20One%20Sheet.pdf.



112. The fourth step of the method in claim 1 also generally recites, the at least one other server in the second tier comprises a second portion of the content provider's library, and the first portion of the content provider's library is distinct from the second portion. On information and belief, for example, the accused Comcast CDN functionalities have mid-tier (*e.g.*, second-tier) cache servers which are comprised of multiple servers that store portions of a content provider's library (*e.g.*, resources on an origin server). In this example, different servers utilized in the accused Comcast CDN functionalities contain different data from each other, and Apache Traffic Server technical documents describe how it allows administrators to configure what type of data may be stored in cache. For example, one server may contain only non-image data, and another server may contain only image data:[86]

[86] https://docs.trafficserver.apache.org/en/7.1.x/admin-guide/configuration/cache-basics.en.html#caching-cookied-objects; https://www.comcasttechnologysolutions.com/sites/default/files/2017-10/CDN%20One%20Sheet.pdf.

Caching Cookied Objects

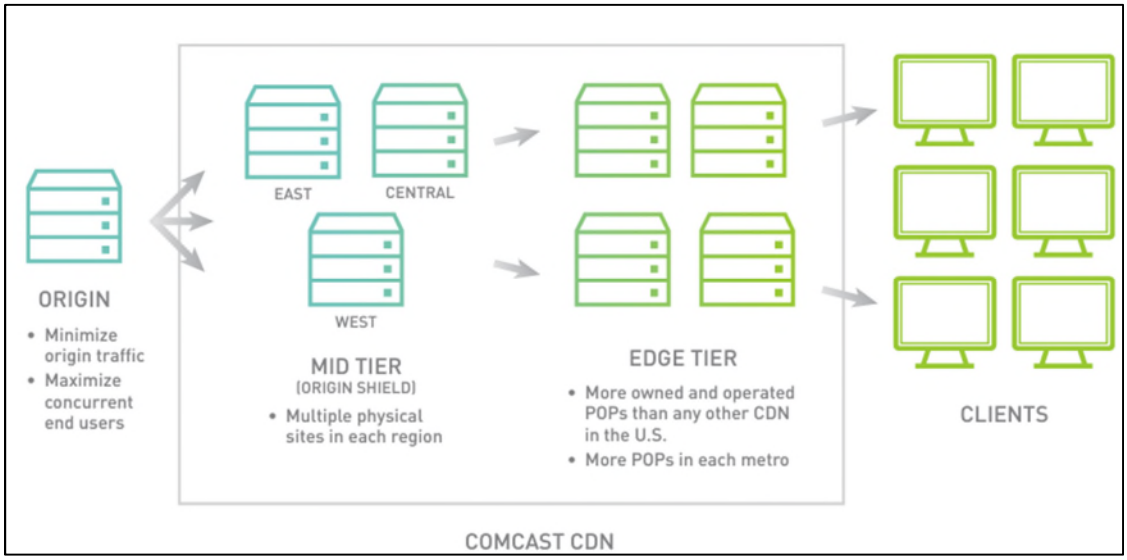
By default, Traffic Server caches objects served in response to requests that contain cookies. This is true for all types of objects including text. Traffic Server does not cache cookied text content because object headers are stored along with the object, and personalized cookie header values could be saved with the object. With non-text objects, it is unlikely that personalized headers are delivered or used.

You can reconfigure Traffic Server to:

- Not cache cookied content of any type.
- Cache cookied content that is of image type only.
- Cache all cookied content regardless of type.

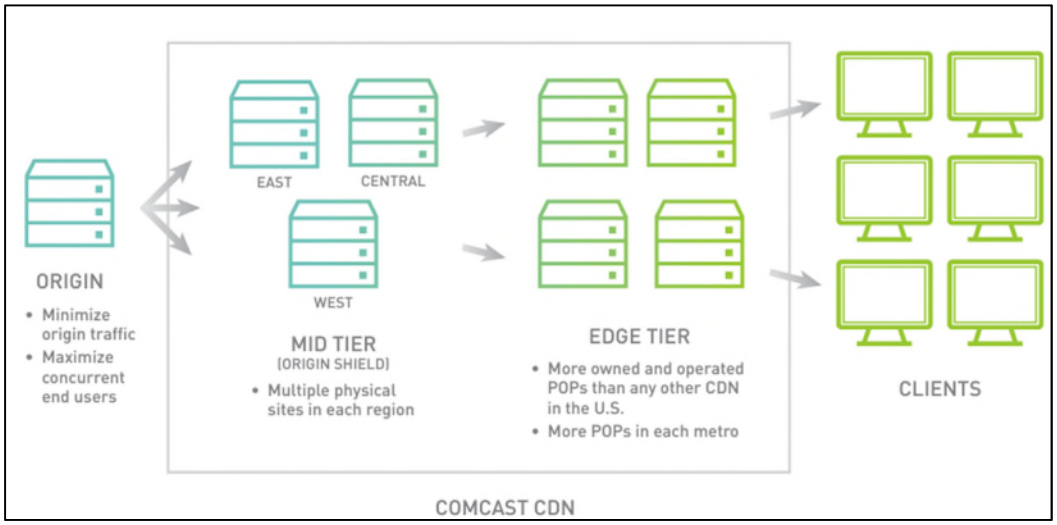
To configure how Traffic Server caches cookied content:

1. Edit `proxy.config.http.cache.cache_responses_to_cookies` in `records.config`.
2. Run the command `traffic_ctl config reload` to apply the configuration changes.



113. The fourth step of the method in claim 1 also generally recites the second tier comprises any intermediate tier between the first tier and an origin server that stores resources associated with the content provider's library. On information and belief, the accused Comcast CDN functionalities include a mid-tier (*e.g.*, second tier) that is an intermediate tier between an origin server (*e.g.*, first tier) which stores resources associated with the content provider's library. For example, Apache Traffic Server source code and technical documents detail how the accused Comcast CDN functionalities first serve requests by edge cache servers, which may pass down

requests to mid-tier cache servers. In this example, all cache servers contain some amount of data from the origin server so they can serve requests on the origin server's behalf.[87]



114. The fifth step of the method in claim 1 generally recites that the second server serve the resource to the client. On information and belief, for example, in the accused Comcast CDN functionalities, upon determining a resource is not popular, an edge-tier server determines not to cache a result. In this example, if no cache action is to be performed, Apache Traffic Server source code details how it does not build a response to be sent to the client but instead creates a tunnel between the responding server (*e.g.*, second server) and the client so that the second server can serve the request.[88]

115. Claim 2 of the '692 Patent depends from the method in claim 1 and further generally recites that portions of the content provider's library are logically partitioned across servers in the second tier. On information and belief, the accused Comcast CDN functionalities use Apache

[87] *See* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *HandleCacheOpenReadMiss( )*, *HandleResponse( )*, and *handle_cache_operation_on_forward_server_response( )* functions); *see also* https://www.comcasttechnologysolutions.com/sites/default/files/2017-10/CDN%20One%20Sheet.pdf.

[88] *See* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *handle_forward_server_connection_open( )* function).

Traffic Server (as a mid-tier cache Server, for instance) which provides configuration to partition cache servers based on specific protocol and size. Accordingly, in this example and on information and belief, the accused Comcast CDN functionalities logically partition portions of the content provider's library across servers in the second tier based on factors such as protocol and size, as discussed in Comcast and Apache Traffic Server technical documents:[89]

- The Comcast CDN, for example, is an active part of the open-source community. Our commercial CDN uses the open-source Apache Traffic Server (ATS) caching engine, and we regularly contribute to this open-source project. We have our own ATS committers on staff.
- Comcast also developed and open sourced a caching control management system called Apache Traffic Control (ATC) that allows users to tie together many ATC caches to form a large-scale CDN. The Traffic Control software is available in GitHub and includes a Traffic Router, Traffic Ops, Traffic Vault, Traffic Monitor, Traffic Stats, and Traffic Portal.

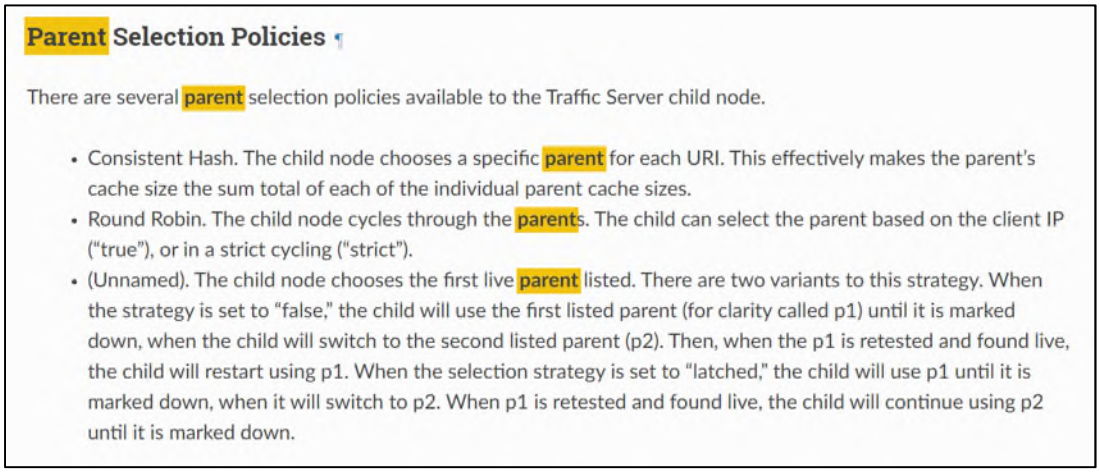
Creating Cache Partitions for Specific Protocols

You can create separate volumes for your cache that vary in size to store content according to protocol. This ensures that a certain amount of disk space is always available for a particular protocol. Traffic Server currently supports only the `http` partition type.

116. Claim 7 of the '692 Patent depends from the method in claim 1 and further generally recites that the step of directing the client to a second server in a second tier is performed in response to a step of using a hash to determine which second server in the second tier stores the first portion of the content provider's library. On information and belief, in one example, the accused Comcast CDN functionalities use Apache Traffic Server to select a server to send a client

[89] https://www.comcasttechnologysolutions.com/sites/default/files/2022-10/Multi-CDN-eBook-2022.pdf; https://docs.trafficserver.apache.org/en/7.1.x/admin-guide/storage/index.en.html#creating-cache-partitions-for-specific-protocols.

request to, select a second tier server, and build a request to that sever for the client requested resource, as explained in Apache Traffic Server source code and technical documents:[90]



Parent Selection Policies

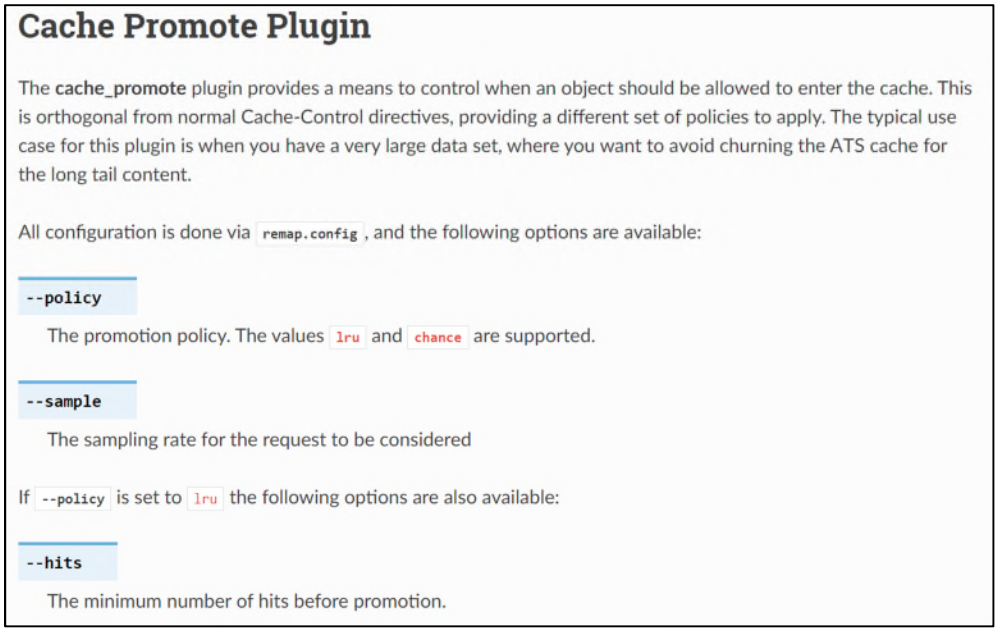
There are several parent selection policies available to the Traffic Server child node.

- Consistent Hash. The child node chooses a specific parent for each URI. This effectively makes the parent's cache size the sum total of each of the individual parent cache sizes.
- Round Robin. The child node cycles through the parents. The child can select the parent based on the client IP ("true"), or in a strict cycling ("strict").
- (Unnamed). The child node chooses the first live parent listed. There are two variants to this strategy. When the strategy is set to "false," the child will use the first listed parent (for clarity called p1) until it is marked down, when the child will switch to the second listed parent (p2). Then, when the p1 is retested and found live, the child will restart using p1. When the selection strategy is set to "latched," the child will use p1 until it is marked down, when it will switch to p2. When p1 is retested and found live, the child will continue using p2 until it is marked down.

117. Claim 8 of the '692 Patent depends from the method in claim 1 and further generally recites that determining whether the resource is popular comprises determining whether a current popularity value for the resource exceeds a popularity threshold. On information and belief, in one example, the accused Comcast CDN functionalities use Apache Traffic Server to implement a LRU ("Last Recently Used") policy in the Cache Promote Plugin, which specifies the only cache content that has reached a certain popularity threshold. In this specific example, Apache Traffic Control source code and technical documents detail how this amount of recorded hits on a cache entry is checked against the hit threshold to evaluate whether to promote a cache entry.[91]

[90] *See* trafficserver/src/proxy/http/HttpTransact.cc (detailing the *HandleCacheOpenRead( )*, *HandleCacheOpenReadMiss( )*, *find_server_and_update_current_info( )*, and *handle_cache_operation_on_forward_server_response( ) functions); see also* trafficserver/src/proxy/ParentSelection.cc (detailing the *findParent( )* function); *see also* trafficserver/src/proxy/ParentConsistentHash.cc (detailing the *selectParent( )* function); *see also* https://docs.trafficserver.apache.org/en/latest/admin-guide/configuration/hierarchical-caching.en.html.

[91] *See* trafficserver/plugins/cache_promote/lru_policy.cc (detailing the *doPromote( )* function); https://docs.trafficserver.apache.org/en/7.1.x/admin-guide/plugins/cache_promote.en.html.



Cache Promote Plugin

The **cache_promote** plugin provides a means to control when an object should be allowed to enter the cache. This is orthogonal from normal Cache-Control directives, providing a different set of policies to apply. The typical use case for this plugin is when you have a very large data set, where you want to avoid churning the ATS cache for the long tail content.

All configuration is done via `remap.config`, and the following options are available:

`--policy`

The promotion policy. The values `lru` and `chance` are supported.

`--sample`

The sampling rate for the request to be considered

If `--policy` is set to `lru` the following options are also available:

`--hits`

The minimum number of hits before promotion.

118. Claim 15 of the '692 Patent claims a method for delivering content in a CDN comprising a plurality of tiers of servers, including a first tier and second tier of servers. Several steps of the method in claim 15 mirror the steps of claim 1, and the examples of how the accused Comcast CDN functionalities perform those steps are described above. The second step of claim 15, recites selectively redirecting the request from the client to a second tier server if the resource requested by a client is not available at the first server or at a peer of the first server. On information and belief, in one example, the accused Comcast CDN functionalities use Apache Traffic Server to determine if a resource is not available at the first server or at a peer of that server, and then to selectively redirect the client request to a second tier server. In this example, on information and belief, Apache Traffic Control source code details how, when a cache miss occurs, another server is located—such as a parent proxy or the origin server—to retrieve the resource from. Moreover, the Apache Traffic Control source code details how, once the edge server finds another server to fetch the requested resource from, a request for the resource is sent to the other server, which subsequently handles the request and updates the cache with the requested resource

and build a response containing the resource to the client.[92] Further to this example, the Apache Traffic Server source code provides that if content is determined to not be popular, the Cache Promote Plugin determines if a plugin has decided not to cache the response; and, if it has not, when a parent proxy server or origin server responds to the request, a tunnel is created between the responding server and the client so that the other server can serve the request.[93]

PRAYER FOR RELIEF

Wherefore, Plaintiff requests entry of judgment in its favor and against Comcast as follows:

A. Judgment that Comcast has directly infringed one or more claims of each of the Asserted Patents pursuant to 35 U.S.C. § 271;

B. An award of damages to compensate Plaintiff for Comcast's infringement, including damages pursuant to 35 U.S.C. § 284, as well as prejudgment and post-judgment interest;

C. An award of costs and expenses in this action, including an award of Plaintiff's reasonable attorneys' fees pursuant to 35 U.S.C. § 285;

D. A permanent injunction restraining and enjoining Comcast, and its respective officers, agents, servants, employees, attorneys, and those persons in active concert or participation with Comcast who receive actual notice of the order by personal service or otherwise, from any further sales or use of their infringing products and/or services and any other infringement of the Asserted Patents;

E. A finding that this is an exceptional case and ordering Comcast to pay Plaintiff's costs of suit and attorneys' fees; and

---

[92] *See* trafficserver/src/proxy/http/HttpTransact.cc (describing the *HandleCacheOpenRead( ), HandleCacheOpenReadMiss( ), find_server_and_update_current_info( ), HandleResponse( ),* and *handle_cache_operation_on_forward_server_response( )* functions); *see also* trafficserver/src/proxy/ParentSelection.cc (describing the *findParent( )* function); *see also* trafficserver/src/proxy/ParentConsistentHash.cc (describing the *selectParent( )* function).

[93] *See* trafficserver/plugins/cache_promote/cache_promote.cc (describing the *Cont_handle_policy( )* function); *see also* trafficserver/src/proxy/http/HttpTransact.cc (describing the *HandleCacheOpenReadMiss( ), find_server_and_update_current_info( ), HandleResponse( ), handle_forward_server_connection_open( ),* and *handle_cache_operation_on_forward_server_response( )* functions).

F. Any such other and further relief as the Court may deem just, proper, and equitable under the circumstances.

JURY DEMAND

Plaintiff respectfully demands a trial by jury on all claims and issues so triable.

Dated: November 1, 2024

Respectfully submitted,

By: /s/ *Robert H. Reckers*
Robert H. Reckers
Attorney-in-Charge
Texas Bar No. 24039520
Andrew M. Long
Texas Bar No. 24123079
Cesar A. Udave
Texas Bar No. 24128501
SHOOK, HARDY & BACON L.L.P.
600 Travis Street, Suite 3400
Houston TX 77002
Phone: 713.227.8008
Fax: 713-227-9508
rreckers@shb.com
andrewmlong@shb.com
cudave@shb.com

B. Trent Webb
Max McGraw (*pro hac forthcoming*)
Mary J. Peal (*pro hac forthcoming*)
SHOOK, HARDY & BACON L.L.P.
2555 Grand Blvd.
Kansas City, MO 64108
Phone: 816.474.6550
Fax: 816.421.5547
BWEBB@shb.com
MMCGRAW@shb.com
MPEAL@shb.com

Attorney for Defendant
Sandpiper CDN, LLC